facebook

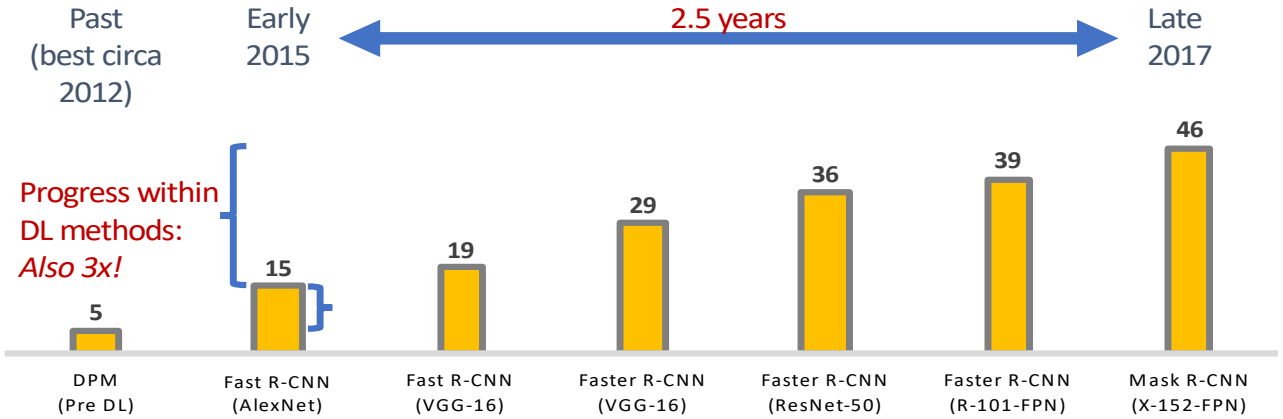
Artificial Intelligence Research

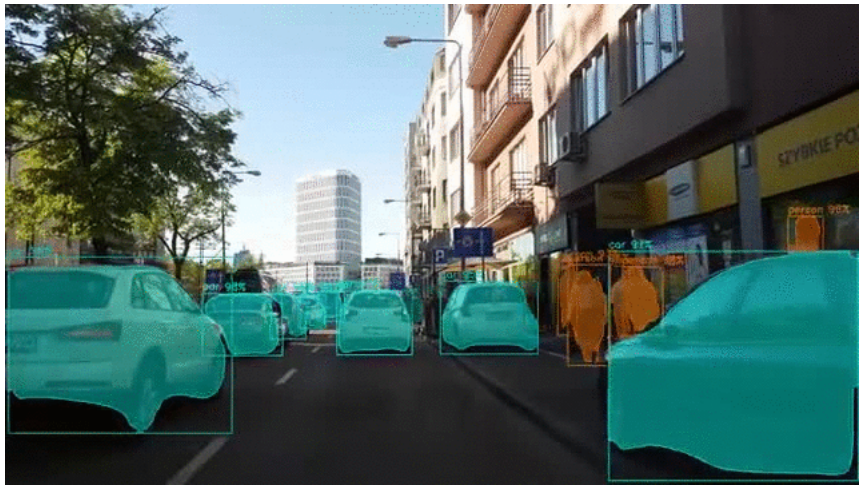
Object Understanding Beyond 2D

Georgia Gkioxari


2D Object Recognition

COCO Object Detection Average Precision (%)





Source: @github/karolmajek

BUT

- 2D object recognition assumes the world is 2D

- 3D properties of the world are largely ignored e.g.
  - Object shapes
  - Scene layout

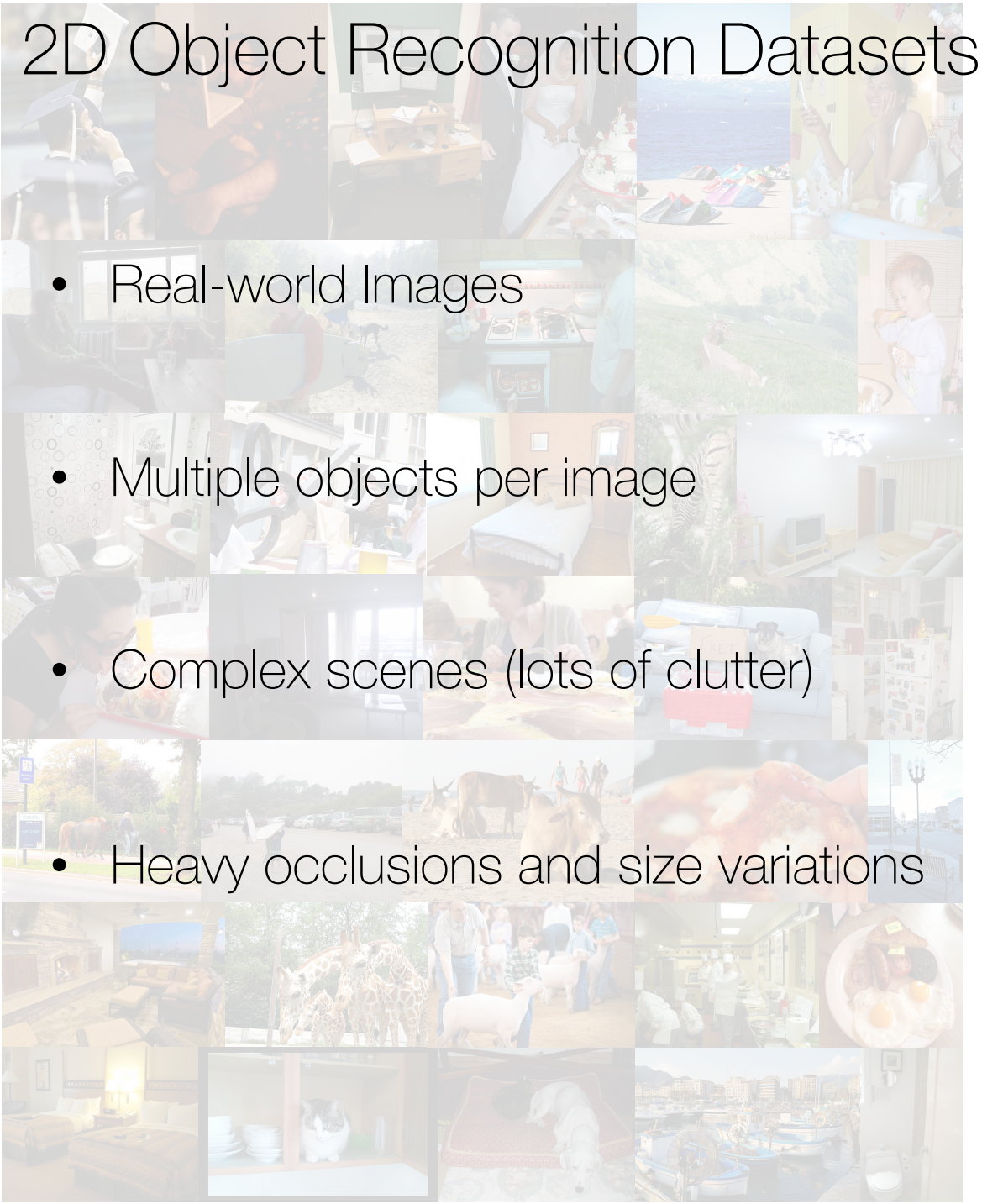
3D Object Recognition


2D Object Recognition


3D Object Recognition

2D Object Recognition Datasets

- Real-world Images
- Multiple objects per image
- Complex scenes (lots of clutter)
- Heavy occlusions and size variations



3D Object Recognition Datasets

- Far from real-world images
- One single object centered in the image
- Blank or simple background (no clutter)
- No occlusion or size variations

2D Object Recognition Models

- Joint object detection, segmentation & pose prediction
- Single model for all object categories
- Models designed to capture diverse appearances
- Occlusion & multiple object confusions need to be tackled
- Reporting metrics is necessary for paper acceptance

3D Object Recognition Models

- Perfect object detector is assumed. Imperfect and erroneous recognition is ignored
- Commonly, one model per object category
- Commonly, shape priors are assumed, e.g. genus 0 shapes or start from mean shapes
- Models assume there is no occlusion or multiple objects. Hard cases are ignored (`occlusion == 0` in dataset selection!)
- Not much emphasis on metrics. More emphasis on qualitative results

3D Object Recognition Datasets

What we need:

- Real-world images
  - Occlusions, multiple objects, clutter, etc. etc. etc.
- *Aligned* 2D and 3D object annotations
  - Necessary for benchmarking
- Object diversity
  - Variety in categories, shapes, appearance, size

Pix3D[1]

- 10k real world images
- 9 object categories
- ~300 3D models of IKEA furniture aligned to real images

[1] Sun et al, Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling, CVPR 2018

3D Object Recognition Models

What we need:

- Joint 2D and 3D understanding in one model
  - Lift the assumption of perfect object recognition
  - Occlusion, clutter, confusions due to multiple instances should be addressed
  - One model for all object categories, similar to 2D object perception (thus, little room for hand-coded object-specific priors)
- Metrics for comparisons and tracking progress

3D Representations

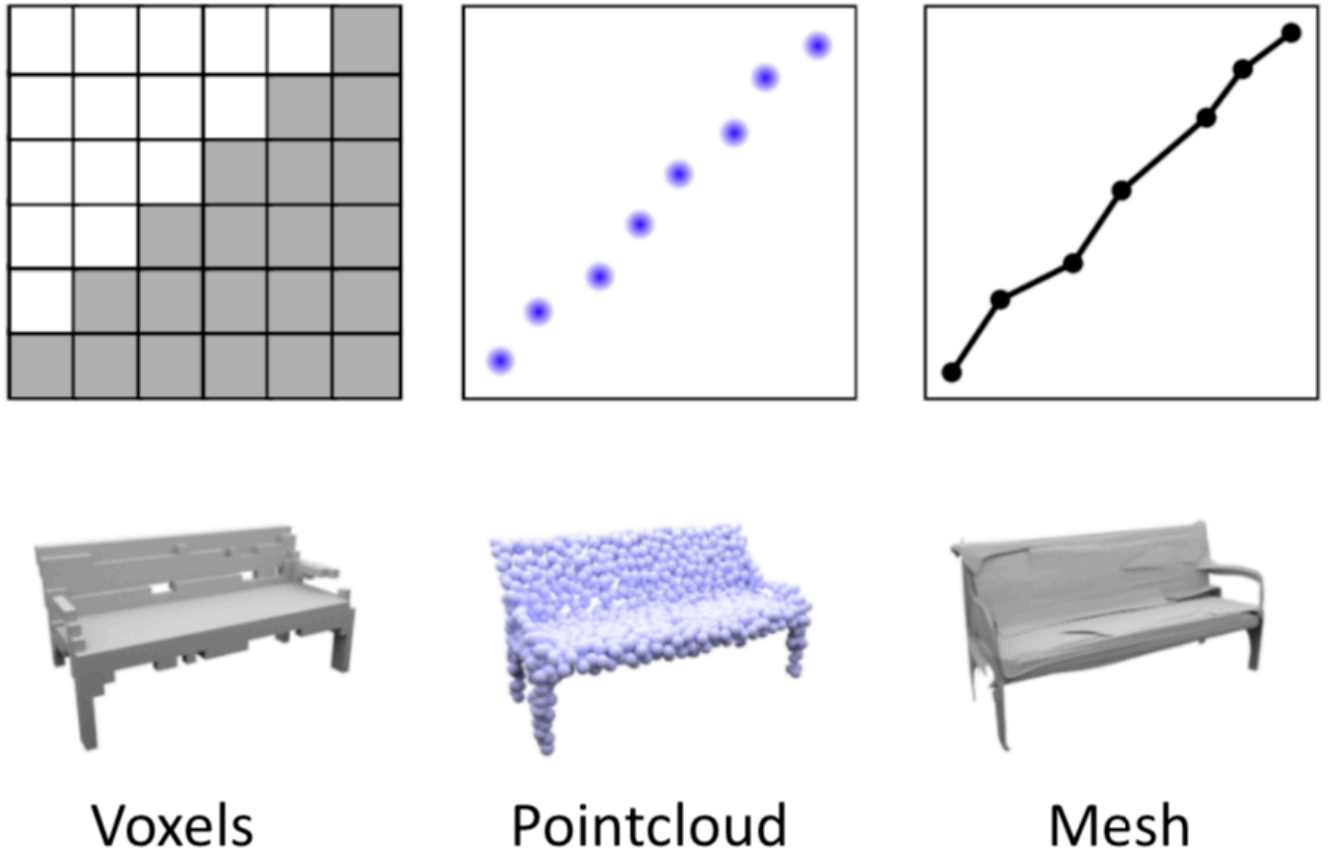


Figure credit: Mescheder et al, Occupancy Networks, CVPR 2019

Voxels

- Represent a shape with a `VxVxV` grid of occupancies
- Just like segmentation masks in Mask R-CNN but in 3D!
- Conceptually simple: just a 3D grid!
- Need high resolution to capture fine structures
- Scaling to high resolutions is non trivial

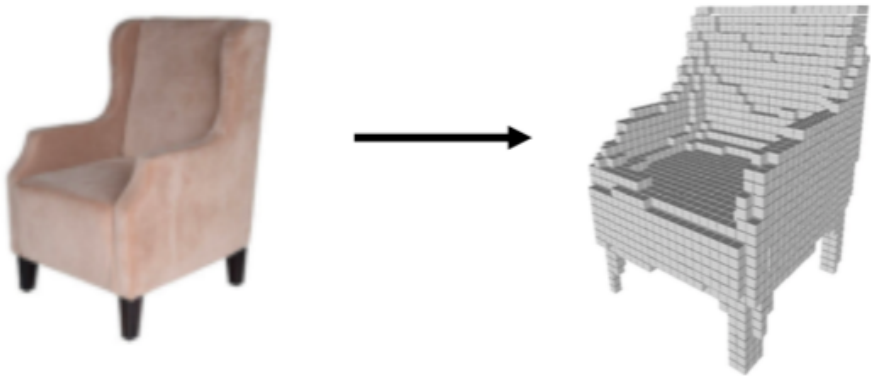


Figure credit: Choy et al, 3D-R2N2, ECCV 2016

Point Clouds

- Represent a shape as set of P points in 3D space
- Requires new architectures and losses (different than grids)
- Can represent fine structures without huge number of points
- Doesn't explicitly represent the surface of the shape: extracting a mesh for rendering or other applications requires post-processing

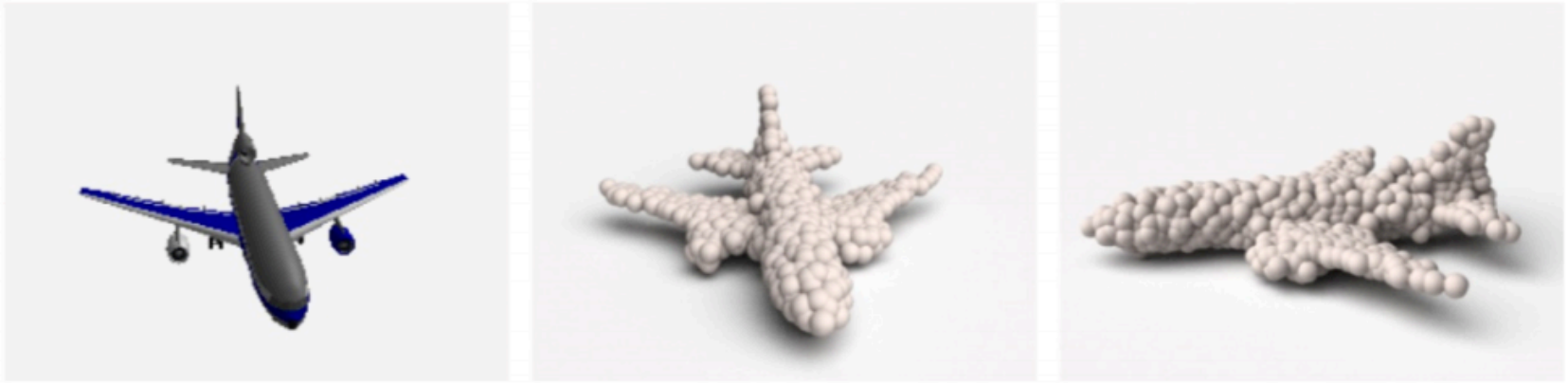
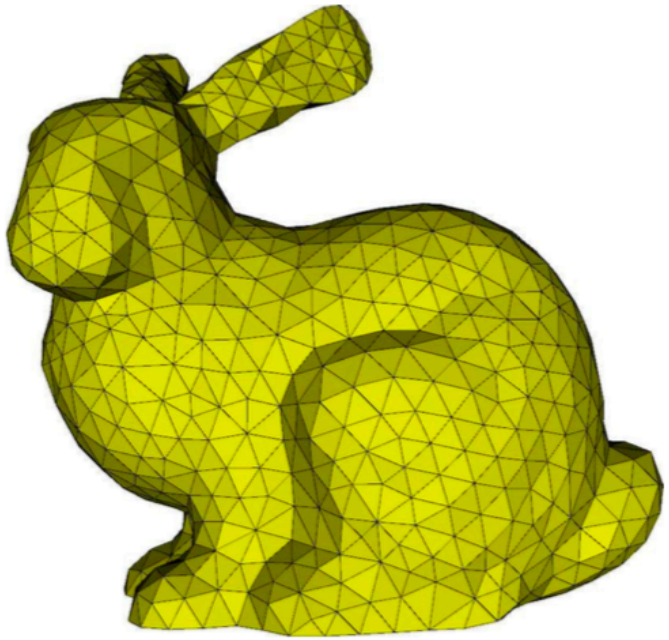


Figure credit: Fan et al, PSG, CVPR 2017

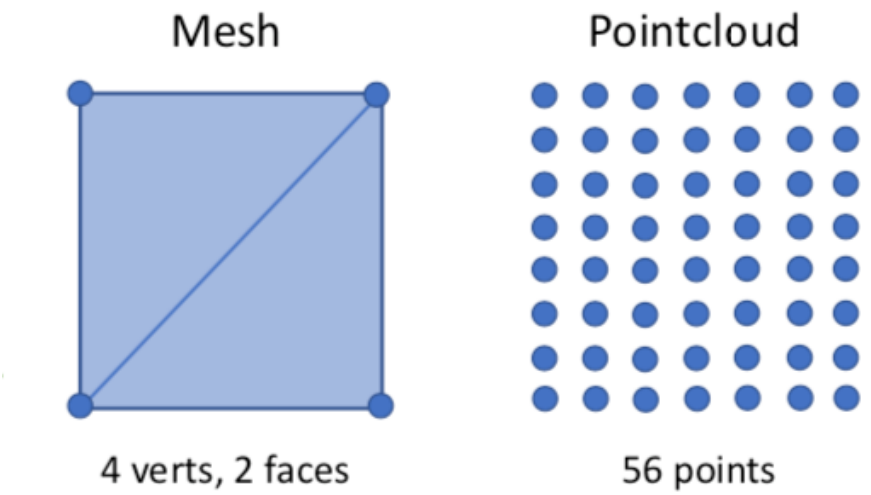
Mesh

- Represent a 3D shape as a set of triangles in 3D space
- **Vertices**: `Vx3` tensor giving real-valued positions in 3D space
- **Faces**: `Fx3` tensor giving indices into the vertices which form triangles
- Standard representation for graphics
- Explicitly represents 3D shapes



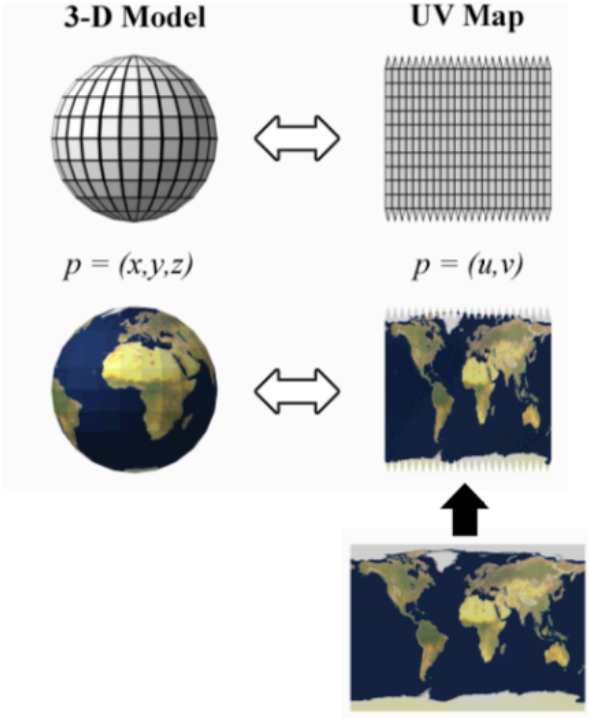
Mesh

- Represent a 3D shape as a set of triangles in 3D space
- **Vertices**: Vx3 tensor giving real-valued positions in 3D space
- **Faces**: Fx3 tensor giving indices into the vertices which form triangles
- Standard representation for graphics
- Explicitly represents 3D shapes
- Adaptive: Can represent surfaces efficiently, can allocate verts to areas with fine detail



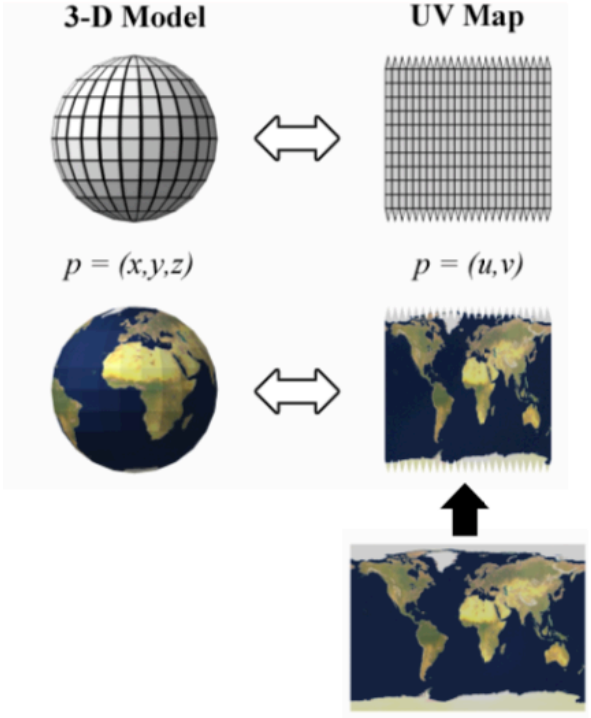
Mesh

- Represent a 3D shape as a set of triangles in 3D space
- **Vertices**: Vx3 tensor giving real-valued positions in 3D space
- **Faces**: Fx3 tensor giving indices into the vertices which form triangles
- Standard representation for graphics
- Explicitly represents 3D shapes
- Adaptive: Can represent surfaces efficiently, can allocate verts to areas with fine detail
- Can attach data to vertices: e.g. texture



Mesh

- Represent a 3D shape as a set of triangles in 3D space
- **Vertices**: Vx3 tensor giving real-valued positions in 3D space
- **Faces**: Fx3 tensor giving indices into the vertices which form triangles
- Standard representation for graphics
- Explicitly represents 3D shapes
- Adaptive: Can represent surfaces efficiently, can allocate verts to areas with fine detail
- Can attach data to vertices: e.g. texture
- Nontrivial to process with neural networks



Cameras

Cameras: Canonical

Canonical coordinates: Predict 3D shape in a canonical system regardless of the viewpoint in the input

Cameras: Canonical vs View

Canonical coordinates: Predict 3D shape in a canonical system regardless of the viewpoint in the input

View coordinates: Predict the 3D shape aligned to the viewpoint of the camera

Cameras: Canonical vs View

Canonical coordinates: Predict 3D shape in a canonical system regardless of the viewpoint in the input

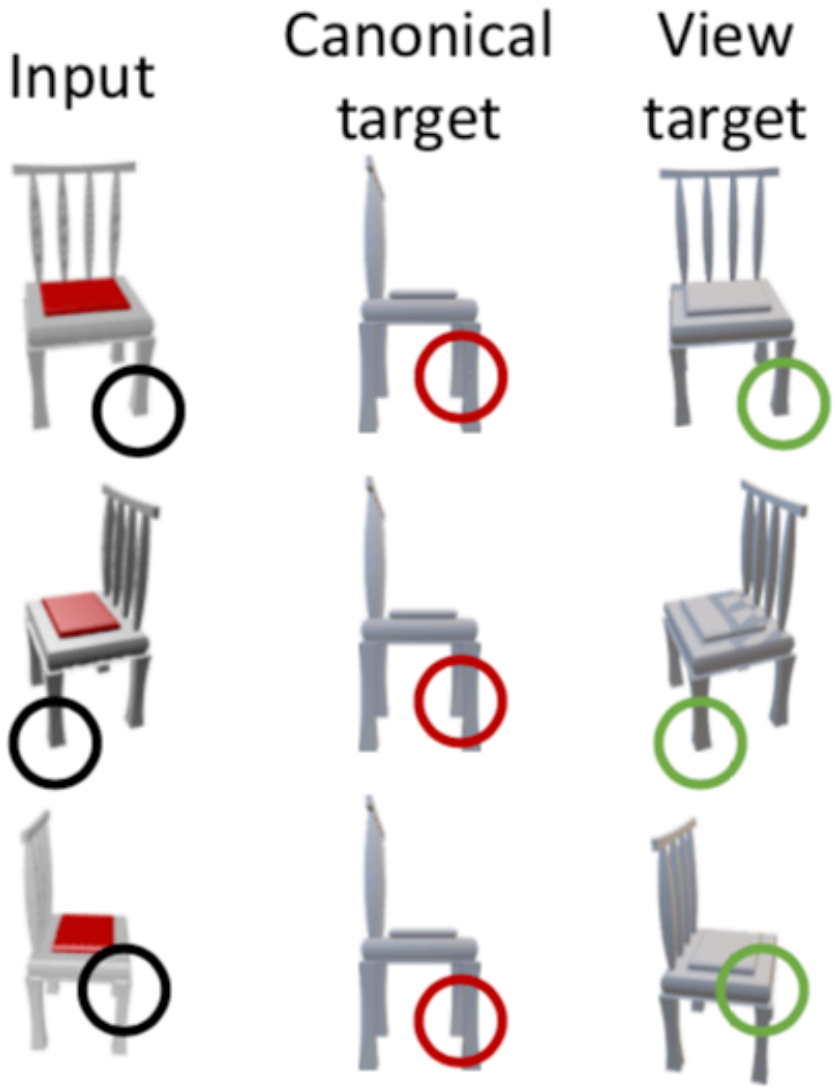
View coordinates: Predict the 3D shape aligned to the viewpoint of the camera

Many papers predict in canonical: easier to load data

Cameras: Canonical vs View

Problem: Canonical view breaks the “principle of feature alignment”: Predictions should be aligned to inputs

View coordinates maintain alignment between inputs and outputs



Mesh R-CNN

Input image

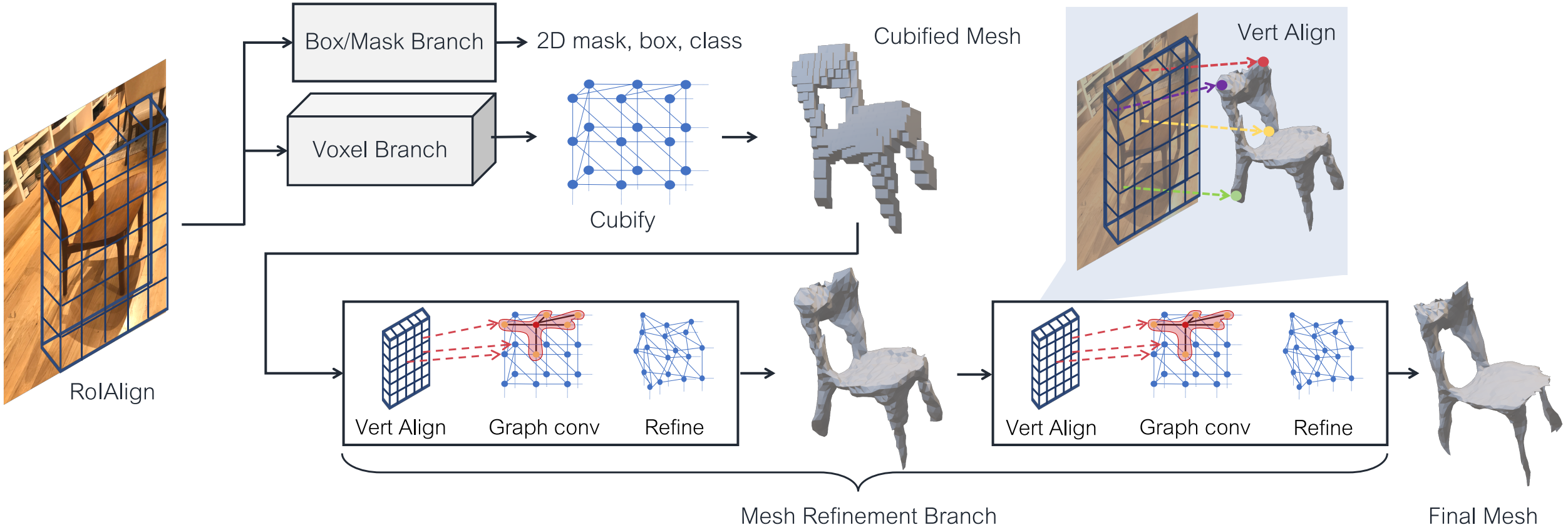
2D object recognition

3D object voxels

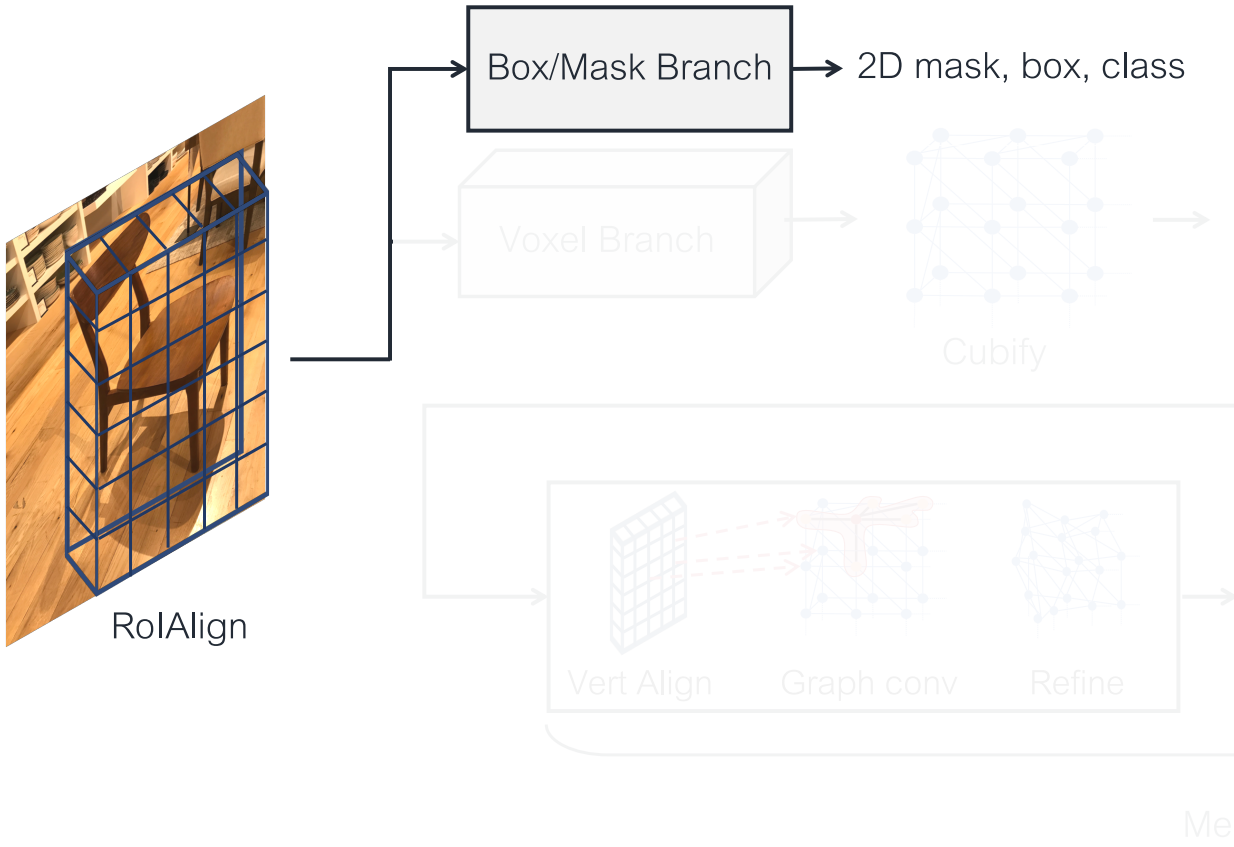
chair

3D object meshes

Mesh R-CNN


Box/Mask Branch
2D mask, box, class
Voxel Branch
Cubify
Cubified Mesh
Vert Align
RoIAlign
Vert Align
Graph conv
Refine
Vert Align
Graph conv
Refine
Mesh Refinement Branch
Final Mesh

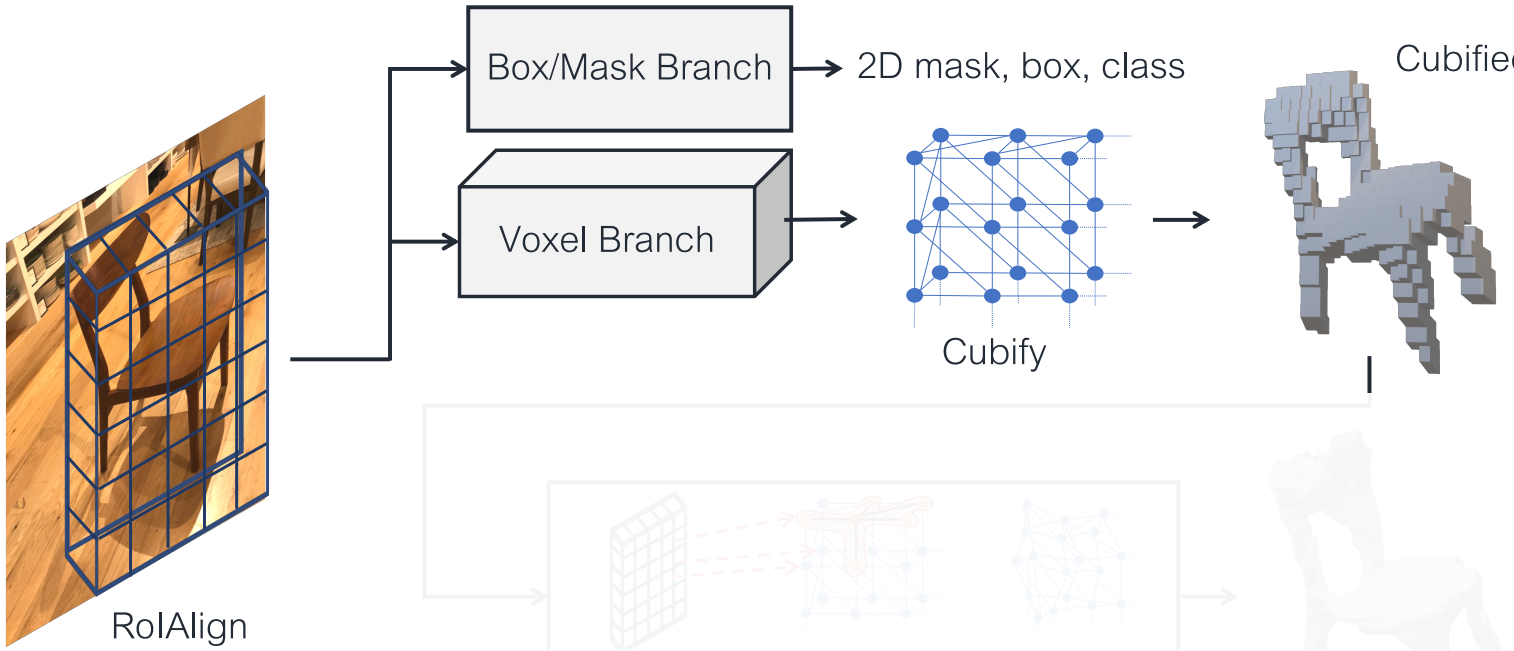
Mesh R-CNN



2D Object Recognition:

- Localizes **all** objects in an image
- Classifies them into object categories

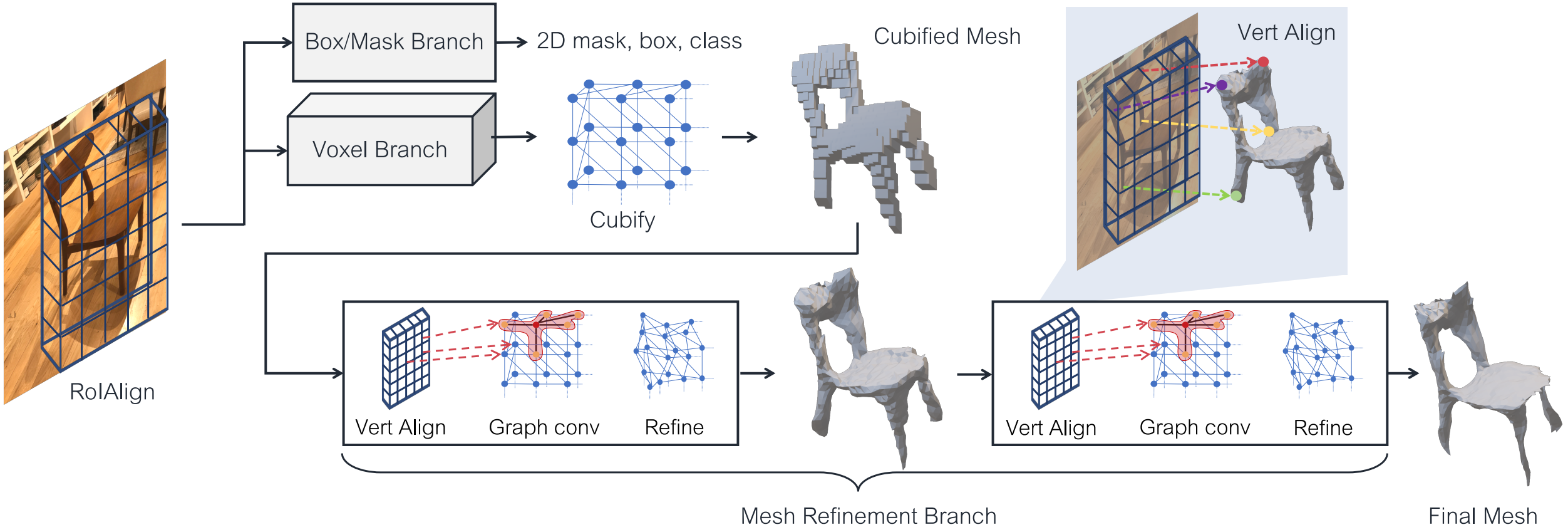
Mesh R-CNN



3D voxels for **each** detected object:

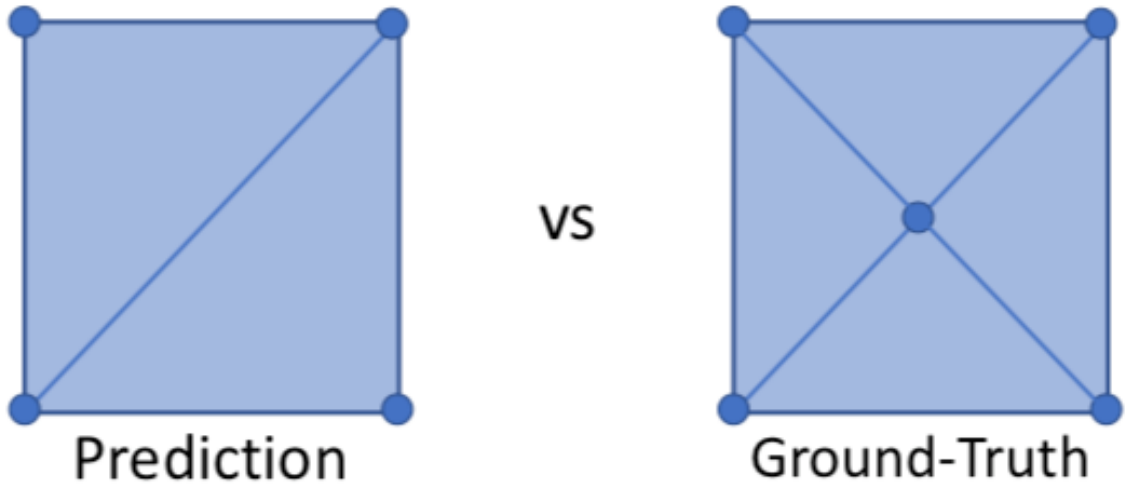
- Captures diverse instance-specific appearance
- In view coordinates
- But voxels are chunky and don't capture fine structures

Mesh R-CNN


Box/Mask Branch
2D mask, box, class
Voxel Branch
Cubify
Cubified Mesh
Vert Align
RoIAlign
Vert Align
Graph conv
Refine
Vert Align
Graph conv
Refine
Mesh Refinement Branch
Final Mesh

Loss Functions

The same shape can be represented with different meshes – how can we define a loss between predicted and ground truth mesh?

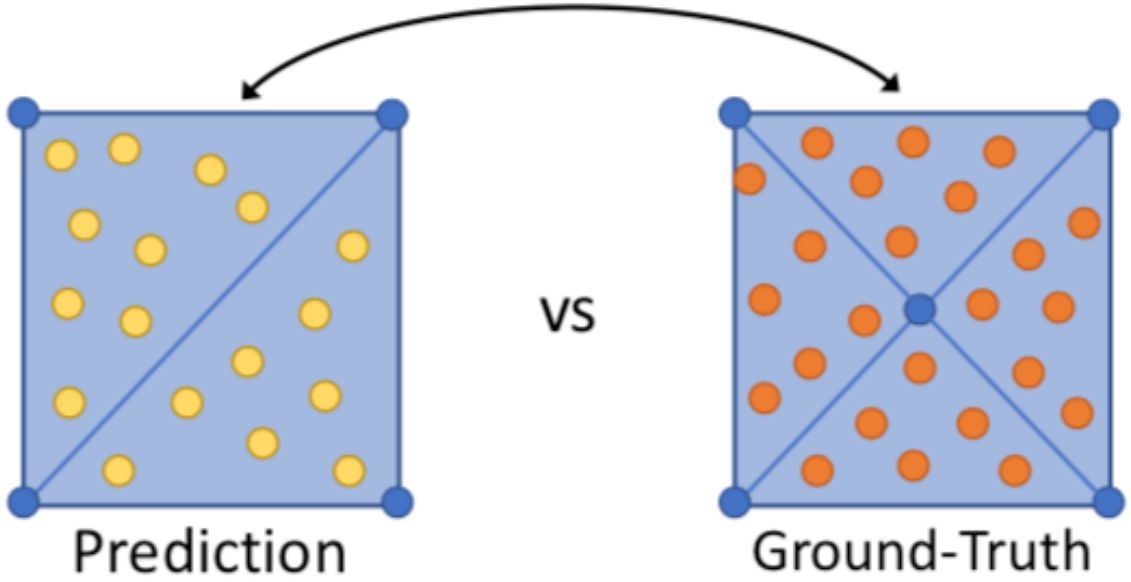


Loss Functions

The same shape can be represented with different meshes – how can we define a loss between predicted and ground truth mesh?

Idea: Convert the meshes into point clouds

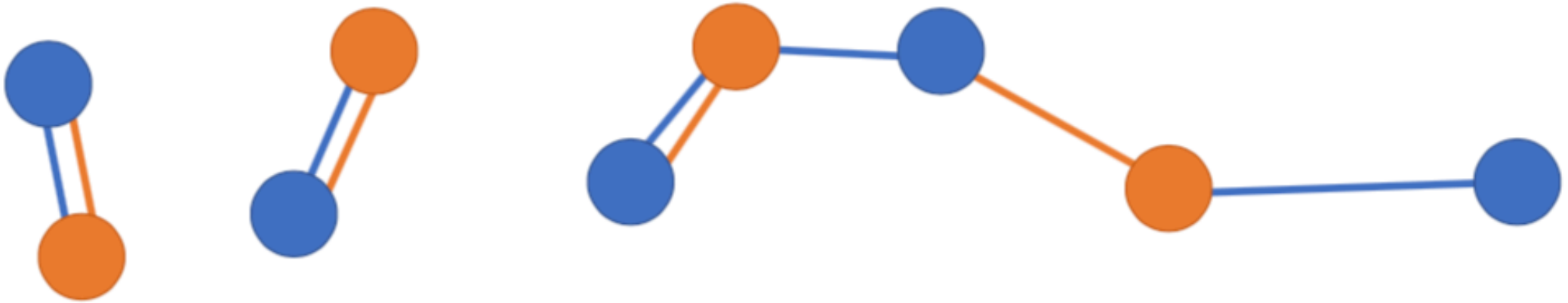
Sample points from the faces of the predicted mesh, online & differentiably!



Sample points from the faces of the ground-truth mesh

Comparing Point Clouds: Chamfer Distance

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2$$



Results on ShapeNet

	Chamfer (↓)	$F1^{\tau}$ (↑)	$F1^{2\tau}$ (↑)
N3MR [25]	2.629	33.80	47.72
3D-R2N2 [5]	1.445	39.01	54.62
PSG [8]	0.593	48.58	69.78
Pixel2Mesh [68]	0.591	59.72	74.19
MVD [55]	-	66.39	-
GEOMetrics [56]	-	67.37	-
Pixel2Mesh [68]‡	0.444	68.94	80.75
Ours (Best)	**0.284**	**75.83**	**86.63**
Ours (Pretty)	0.366	70.70	82.59

Inputs

Predictions

Results on ShapeNet

	Chamfer (↓)	$F1^{\tau}$ (↑)	$F1^{2\tau}$ (↑)
N3MR [25]	2.629	33.80	47.72
3D-R2N2 [5]	1.445	39.01	54.62
PSG [8]	0.593	48.58	69.78
Pixel2Mesh [68]	0.591	59.72	74.19
MVD [55]	-	66.39	-
GEOMetrics [56]	-	67.37	-
Pixel2Mesh [68]‡	0.444	68.94	80.75
Ours (Best)	**0.284**	**75.83**	**86.63**
Ours (Pretty)	0.366	70.70	82.59

Measures the bi-directional distance between two pointclouds

Inputs

Predictions

Results on ShapeNet

	Chamfer (↓)	$F1^{\tau}$ (↑)	$F1^{2\tau}$ (↑)
N3MR [25]	2.629	33.80	47.72
3D-R2N2 [5]	1.445	39.01	54.62
PSG [8]	0.593	48.58	69.78
Pixel2Mesh [68]	0.591	59.72	74.19
MVD [55]	-	66.39	-
GEOMetrics [56]	-	67.37	-
Pixel2Mesh [68]‡	0.444	68.94	80.75
Ours (Best)	**0.284**	**75.83**	**86.63**
Ours (Pretty)	0.366	70.70	82.59

Harmonic mean of the chamfer distance with threshold τ

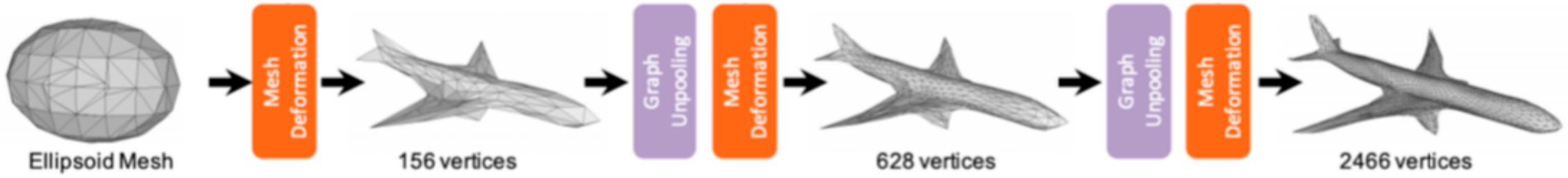
Inputs

Predictions

Prior Work: Pixel2Mesh

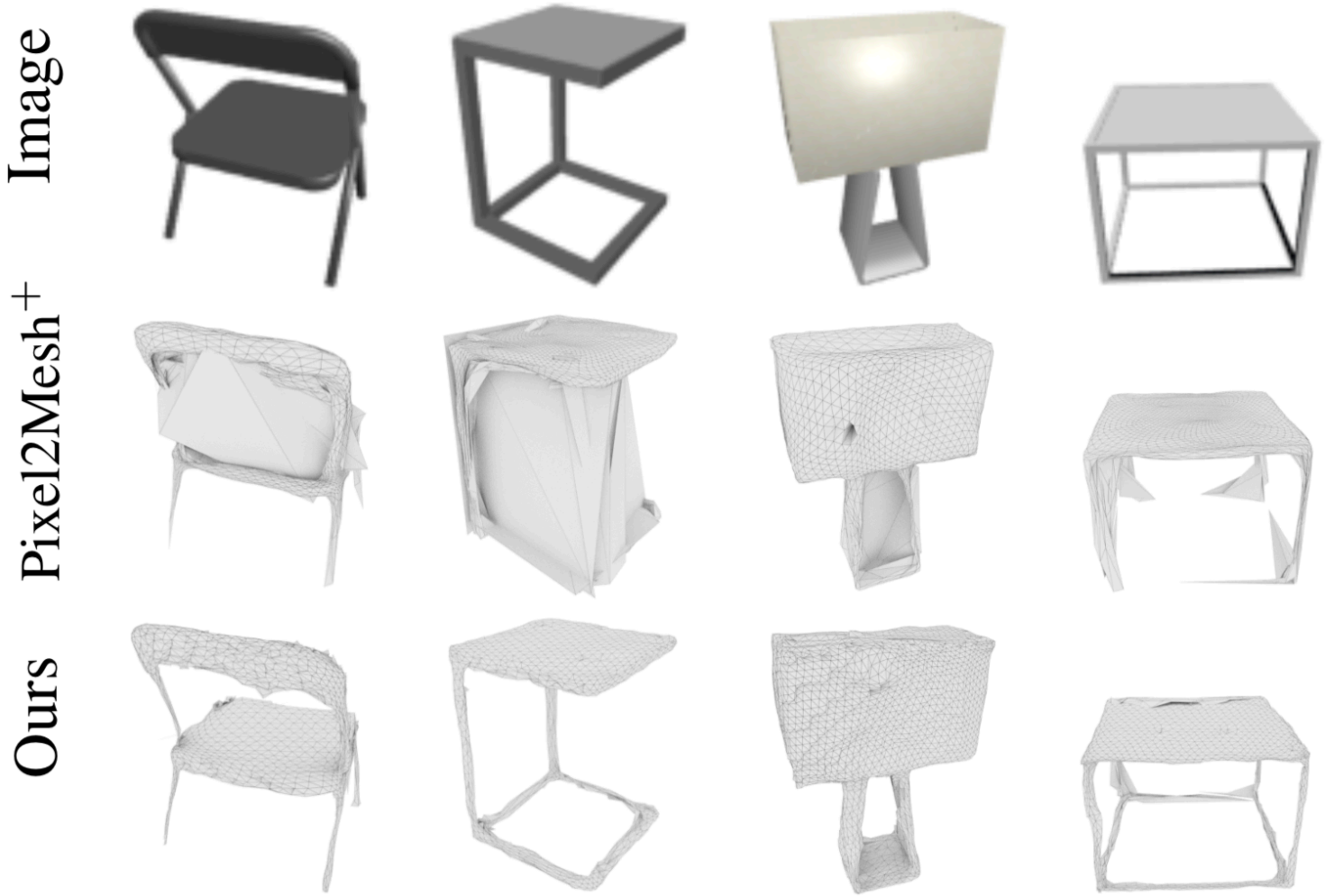
Iterative Refinement of Meshes:

Start from initial ellipsoid mesh, predict offset of vertices via neural network, repeat

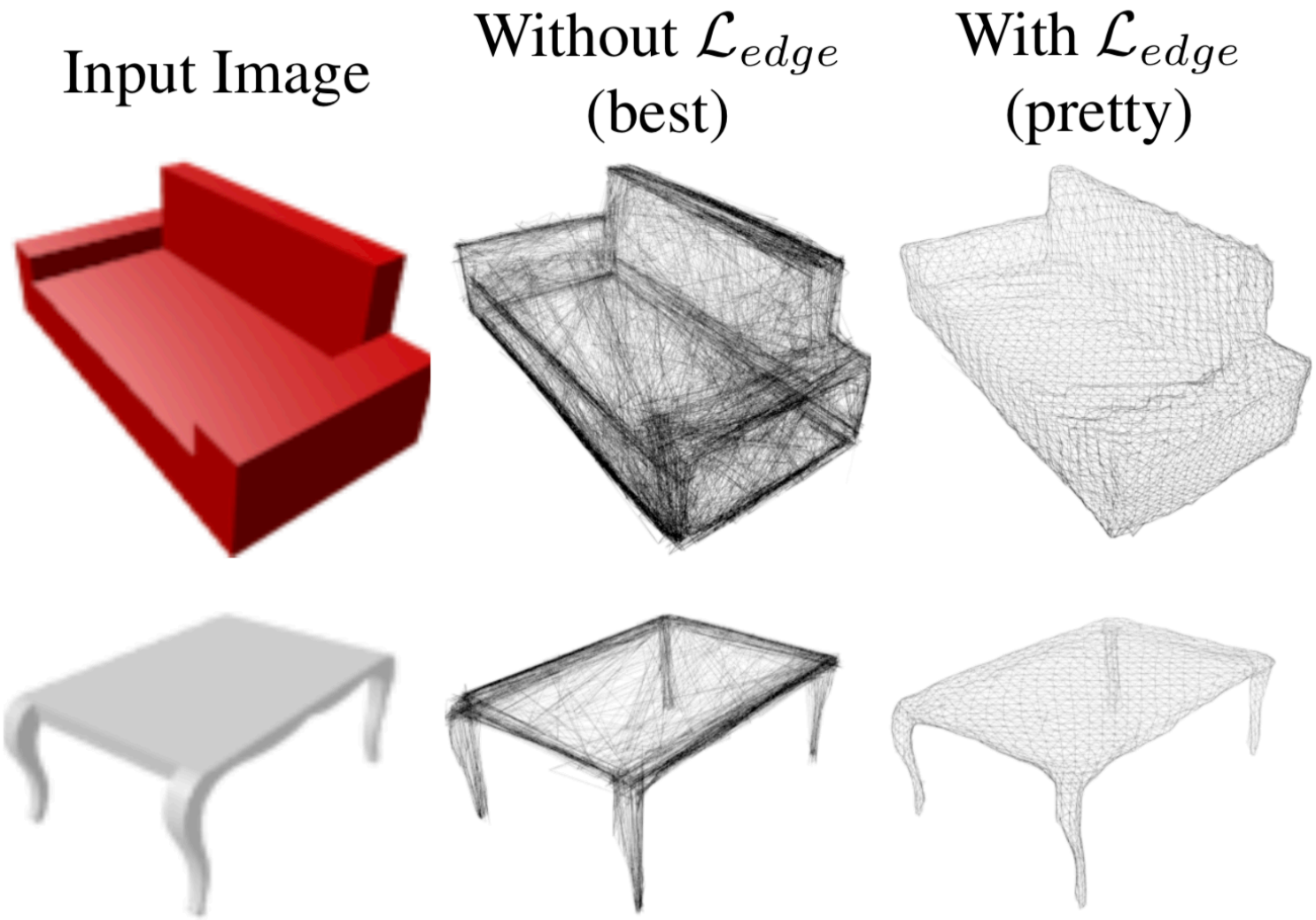


Wang et al, Pixel2Mesh, ECCV 2018

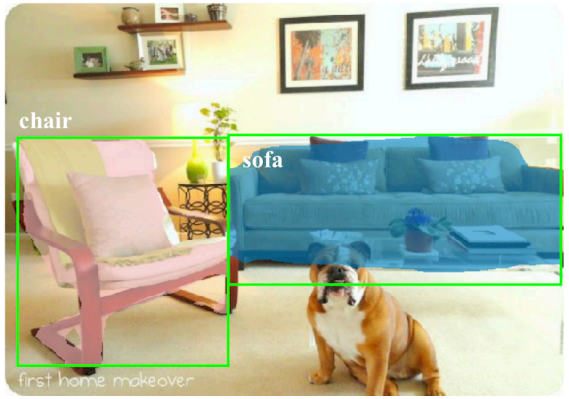
Shape Priors

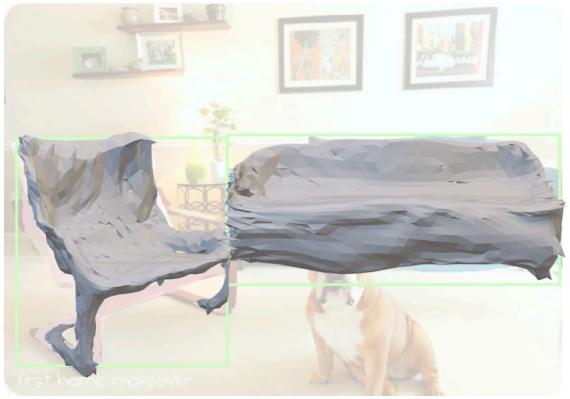


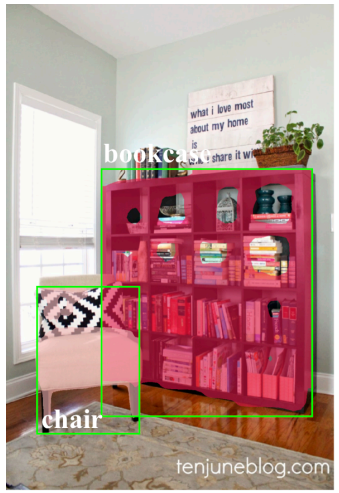
Shape Regularizers



Results on Pix3D







Performance on Pix3D

Detection setting

- $\text{BoxAP}^{0.5}$ measures the AP for 2D box prediction with box IOU at 0.5
- $\text{MaskAP}^{0.5}$ measures the AP for 2D instance mask prediction with mask IOU at 0.5
- $\text{MeshAP}^{0.5}$ measures the AP for 3D instance mesh prediction with F1 at 0.5
  - A prediction is considered to be correct iff
    - The correct class is predicted
    - The F1 score between the predicted and ground truth mesh is above 0.5
    - The prediction is not a duplicate

Results on Pix3D

Random Split S_1:

Images are split randomly into ~7500 train and ~2500 test.

Pix3D $\mathcal{S}_1$	AP^{box}	AP^{mask}	AP^{mesh}	*chair*	*sofa*	*table*	*bed*	*desk*	*bkcs*	*wrdrb*	*tool*	*misc*	$\|V\|$	$\|F\|$
Voxel-Only	92.1	86.6	2.1	0.0	0.3	0.7	0.1	0.5	14.3	3.1	0.0	0.0	$1548_{\pm 506}$	$2855_{\pm 917}$
Pixel2Mesh^{+}	90.4	86.5	42.6	32.5	61.8	46.6	43.1	36.3	57.5	43.9	29.5	**32.4**	$2562_{\pm 0}$	$5120_{\pm 0}$
Sphere-Init	90.4	85.4	42.1	32.8	61.4	49.1	38.4	33.4	53.7	45.7	**40.2**	23.7	$2562_{\pm 0}$	$5120_{\pm 0}$
Mesh R-CNN (ours)	92.3	87.4	**48.4**	**42.6**	**66.9**	**56.6**	**48.2**	**38.6**	**70.0**	**54.2**	36.6	21.8	$1548_{\pm 506}$	$2855_{\pm 917}$
# test instances	2440	2440	2440	1129	398	398	205	148	79	53	11	19		

Results on Pix3D

Shape Split S_2:

Images are split such that a shape seen on test is **not** seen on train

For example,

train for chair

test for chair

Results on Pix3D

Pix3D $\mathcal{S}_1$	AP^{box}	AP^{mask}	AP^{mesh}	*chair*	*sofa*	*table*	*bed*	*desk*	*bkcs*	*wrdrb*	*tool*	*misc*	$\lvert V\rvert$	$\lvert F\rvert$
Voxel-Only	92.1	86.6	2.1	0.0	0.3	0.7	0.1	0.5	14.3	3.1	0.0	0.0	$1548_{\pm 506}$	$2855_{\pm 917}$
Pixel2Mesh^{+}	90.4	86.5	42.6	32.5	61.8	46.6	43.1	36.3	57.5	43.9	29.5	**32.4**	$2562_{\pm 0}$	$5120_{\pm 0}$
Sphere-Init	90.4	85.4	42.1	32.8	61.4	49.1	38.4	33.4	53.7	45.7	**40.2**	23.7	$2562_{\pm 0}$	$5120_{\pm 0}$
Mesh R-CNN (ours)	92.3	87.4	**48.4**	**42.6**	**66.9**	**56.6**	**48.2**	**38.6**	**70.0**	**54.2**	36.6	21.8	$1548_{\pm 506}$	$2855_{\pm 917}$
# test instances	2440	2440	2440	1129	398	398	205	148	79	53	11	19		
Pix3D $\mathcal{S}_2$														
Voxel-Only	66.5	60.1	3.9	0.0	0.0	0.5	0.2	0.8	16.1	0.1	17.0	0.0	$1496_{\pm 437}$	$2758_{\pm 799}$
Pixel2Mesh^{+}	60.4	57.3	22.3	24.9	66.2	14.5	42.6	6.8	20.0	1.2	24.1	0.0	$2562_{\pm 0}$	$5120_{\pm 0}$
Sphere-Init	63.0	58.4	22.0	23.9	68.8	17.5	38.2	5.5	17.9	0.8	25.2	0.0	$2562_{\pm 0}$	$5120_{\pm 0}$
Mesh R-CNN (ours)	63.7	58.9	25.1	27.0	74.3	22.6	38.2	8.6	20.0	1.7	33.2	0.0	$1496_{\pm 437}$	$2758_{\pm 799}$
# test instances	2368	2368	2368	778	506	398	219	205	85	135	22	20		

Results on Pix3D

Pix3D $\mathcal{S}_1$	AP^{box}	AP^{mask}	AP^{mesh}	*chair*	*sofa*	*table*	*bed*	*desk*	*bkcs*	*wrdrb*	*tool*	*misc*	$\|V\|$	$\|F\|$
Voxel-Only	92.1	86.6	2.1	0.0	0.3	0.7	0.1	0.5	14.3	3.1	0.0	0.0	$1548_{\pm 506}$	$2855_{\pm 917}$
Pixel2Mesh^{+}	90.4	86.5	42.6	32.5	61.8	46.6	43.1	36.3	57.5	43.9	29.5	**32.4**	$2562_{\pm 0}$	$5120_{\pm 0}$
Sphere-Init	90.4	85.4	42.1	32.8	61.4	49.1	38.4	33.4	53.7	45.7	**40.2**	23.7	$2562_{\pm 0}$	$5120_{\pm 0}$
Mesh R-CNN (ours)	92.3	87.4	**48.4**	**42.6**	**66.9**	**56.6**	**48.2**	**38.6**	**70.0**	**54.2**	36.6	21.8	$1548_{\pm 506}$	$2855_{\pm 917}$
# test instances	2440	2440	2440	1129	398	398	205	148	79	53	11	19		
Pix3D $\mathcal{S}_2$														
Voxel-Only	66.5	60.1	3.9	0.0	0.0	0.5	0.2	0.8	16.1	0.1	17.0	0.0	$1496_{\pm 437}$	$2758_{\pm 799}$
Pixel2Mesh^{+}	60.4	57.3	22.3	24.9	66.2	14.5	**42.6**	6.8	**20.0**	1.2	24.1	0.0	$2562_{\pm 0}$	$5120_{\pm 0}$
Sphere-Init	63.0	58.4	22.0	23.9	68.8	17.5	38.2	5.5	17.9	0.8	25.2	0.0	$2562_{\pm 0}$	$5120_{\pm 0}$
Mesh R-CNN (ours)	63.7	58.9	**25.1**	**27.0**	**74.3**	**22.6**	38.2	**8.6**	**20.0**	**1.7**	**33.2**	0.0	$1496_{\pm 437}$	$2758_{\pm 799}$
# test instances	2368	2368	2368	778	506	398	219	205	85	135	22	20		

Challenges

Dataset size is still small

- ~10k images in Pix3D, ~300k in COCO
- Scaling to larger datasets is hard

Challenges

3D reconstruction is a recognition problem!

- Occlusions are a big source of errors, similar to 2D recognition
- Fine structures are hard to predict, similar to 2D recognition

Challenges

3D reconstruction is a recognition problem!

- Occlusions are a big source of errors, similar to 2D recognition
- Fine structures are hard to predict, similar to 2D recognition

Challenges: Engineering

PyTorch3d: Pytorch library for deep learning in 3D

- **3D Data structures** (e.g. Meshes, Points) for efficient batching of 3D shapes of diverse topologies and efficient representation conversions

- **3D ops with gradients**
  - Chamfer loss
  - Differentiable mesh sampling
  - Mesh subdivision
  - Graph convolution
  - etc

- **Differentiable renderer**
  - Modular implementation for rasterization, lighting, shading and blending

Batching Meshes

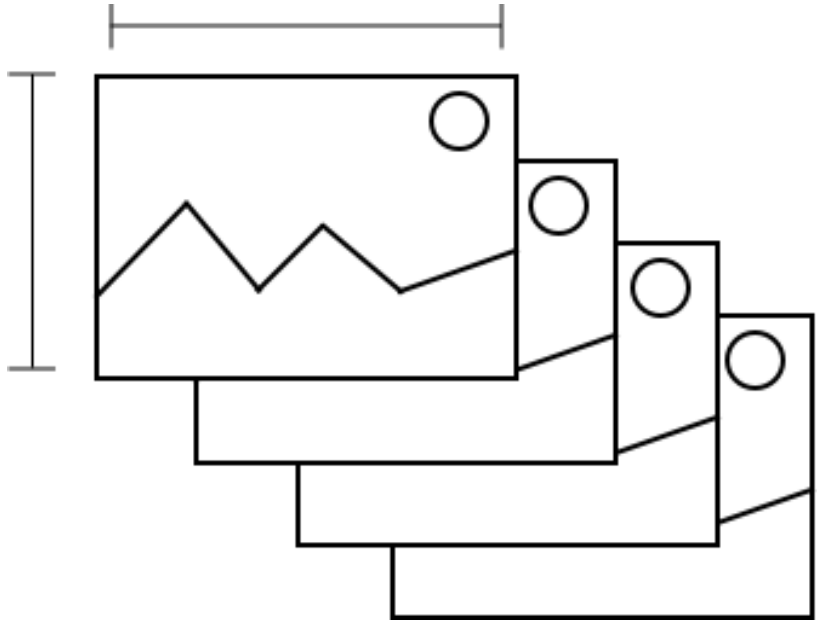
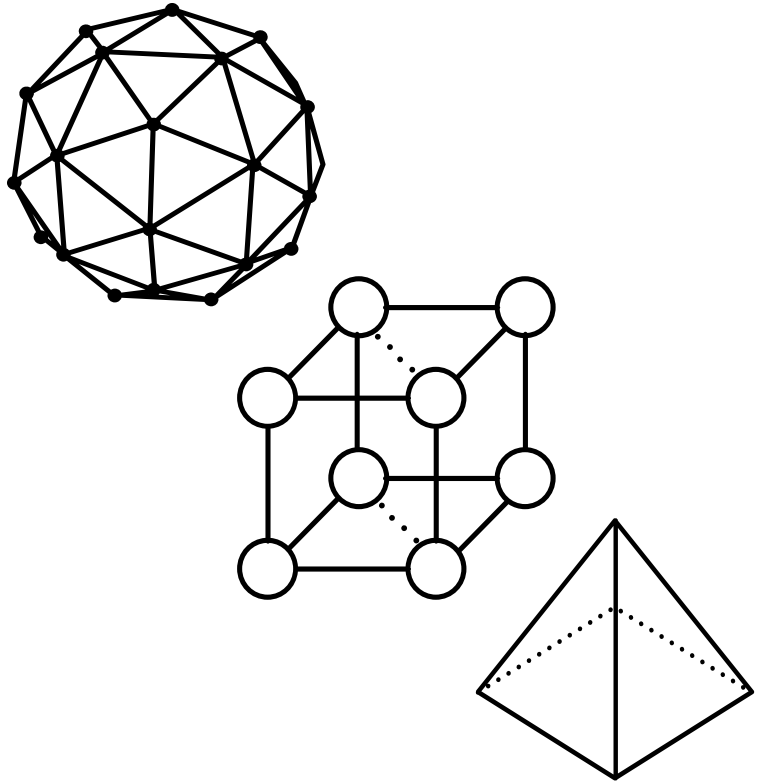


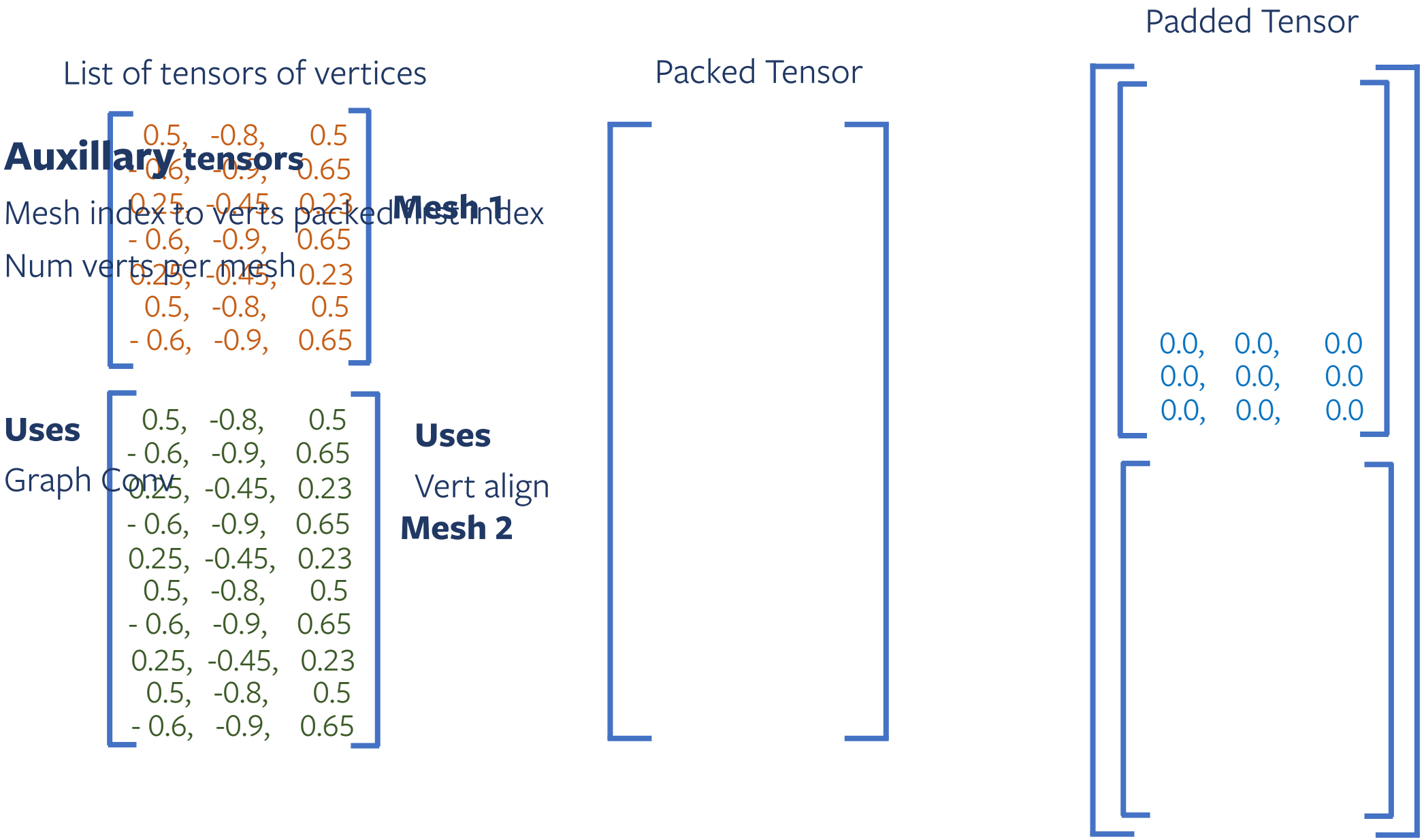
Image: HxWx3 (rgb)

Batch: N x 3 x H x W



Mesh: Vx3 vertices, Fx3 faces

Batch: N x ???

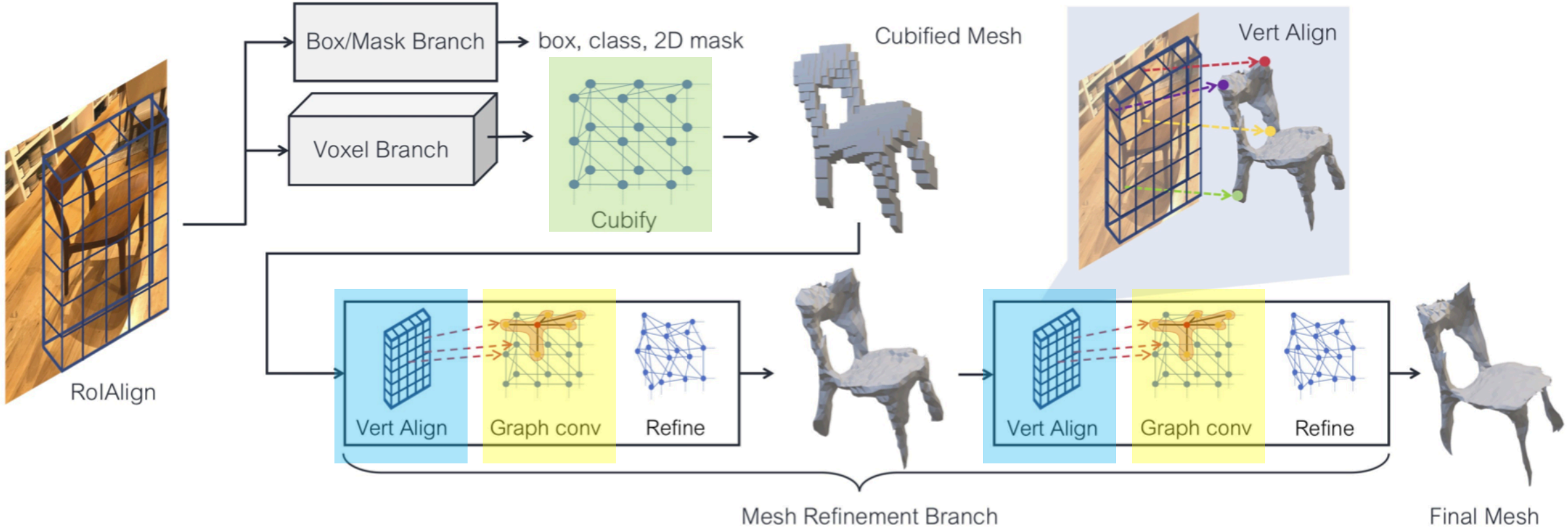

List of tensors of vertices
0.5, -0.8, 0.5
- 0.6, -0.9, 0.65
0.25, -0.45, 0.23
- 0.6, -0.9, 0.65
0.25, -0.45, 0.23
0.5, -0.8, 0.5
- 0.6, -0.9, 0.65
Auxillary tensors
Mesh index to verts packed first index
Num verts per mesh
Mesh 1
Uses
Graph Conv
0.5, -0.8, 0.5
- 0.6, -0.9, 0.65
0.25, -0.45, 0.23
- 0.6, -0.9, 0.65
0.25, -0.45, 0.23
0.5, -0.8, 0.5
- 0.6, -0.9, 0.65
0.25, -0.45, 0.23
0.5, -0.8, 0.5
- 0.6, -0.9, 0.65
Uses
Vert align
Mesh 2
Packed Tensor
Padded Tensor
0.0, 0.0, 0.0
0.0, 0.0, 0.0
0.0, 0.0, 0.0

Mesh R-CNN

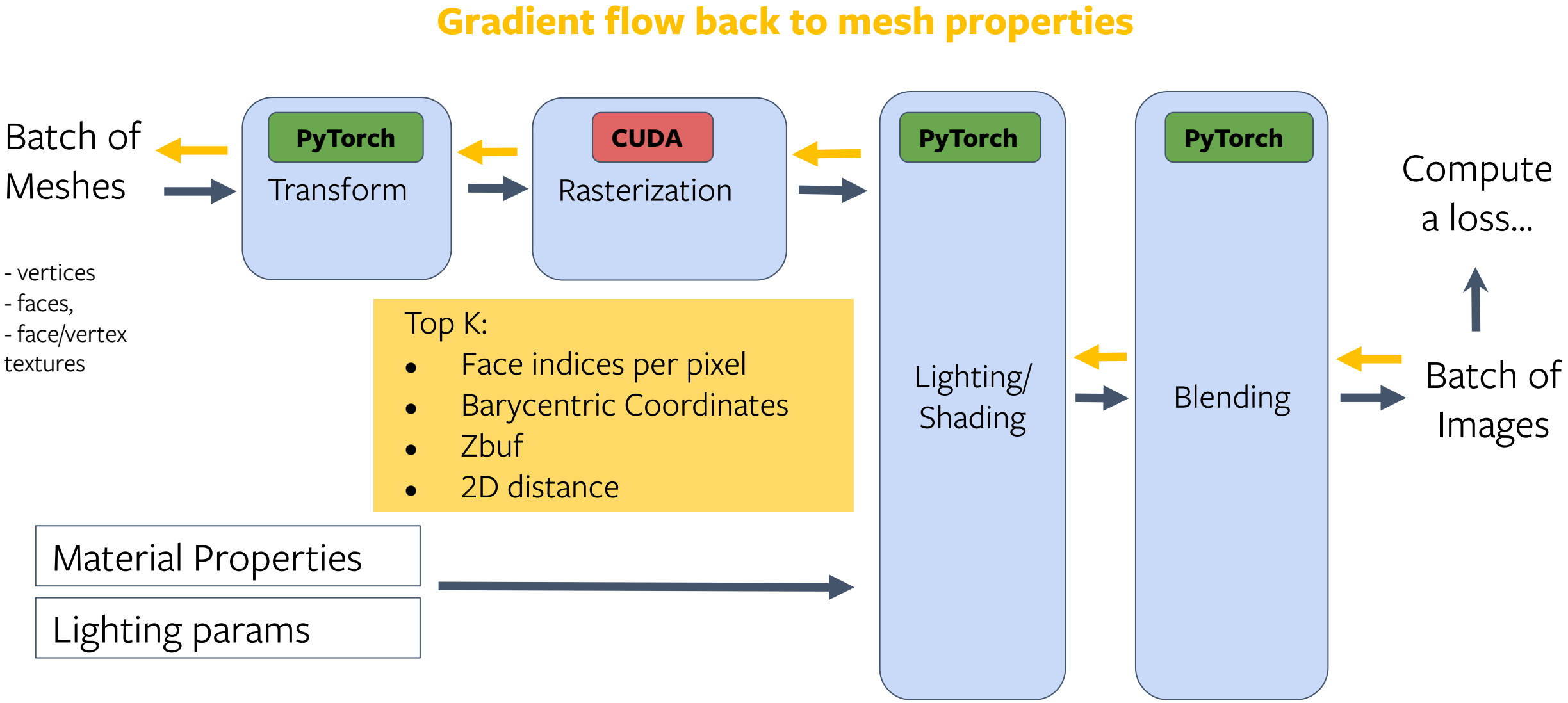
List verts/faces

Padded verts/faces

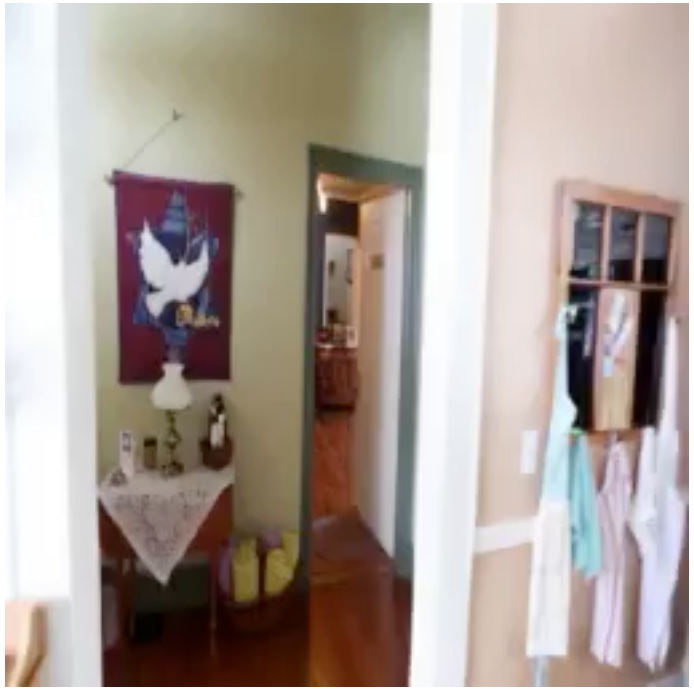
Packed verts/faces



Modular Differentiable Renderer



Novel View Synthesis



Novel View Synthesis

- **3D structure of the scene**: crucial for capturing the relative motion of visible objects under a view transform

- **Semantics**: crucial for synthesizing plausible completions of partially visible objects

Novel View Synthesis: Prior Work

- **Multiple View**: Multiple views of the scene are used as input during test time. This simplifies the problem of 3D understanding (e.g. SfM) and semantics (fewer positions are occluded from more views) [Debevec et al. '96, Fitzgibbon et al. '05, Seitz et al. '06, Zitnick et al. '04]

- **Depth Predictions**: Requires a training dataset of images and ground truth depth. How about domains for which we RGBD cameras fail (e.g. park)? [Eigen et al. '14, Li & Snavely '18, Klaus et al. '19]

- **Im2Im**: Image to image generative models which bypass 3D reasoning [Isola et al. '17, Brock et al. '18]

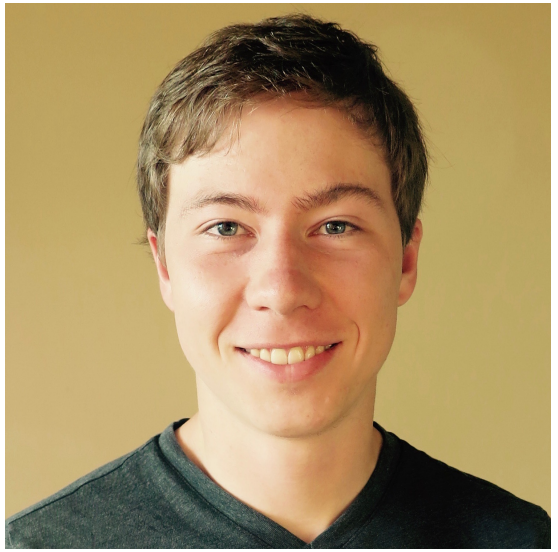
- **3D based approaches**: Learn implicit 3D representations. Mostly emphasize on synthetic datasets, e.g. ShapeNet [Slitzman et al. '19]

End-to-End View Synthesis from a Single Image

Olivia Wiles

Georgia Gkioxari

Rick Szeliski



Justin Johnson

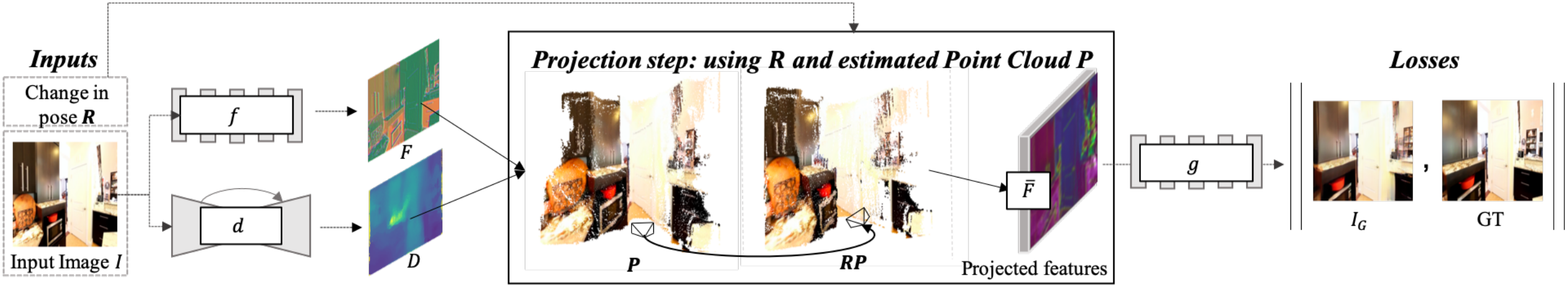
End-to-End View Synthesis from a Single Image

Input Image

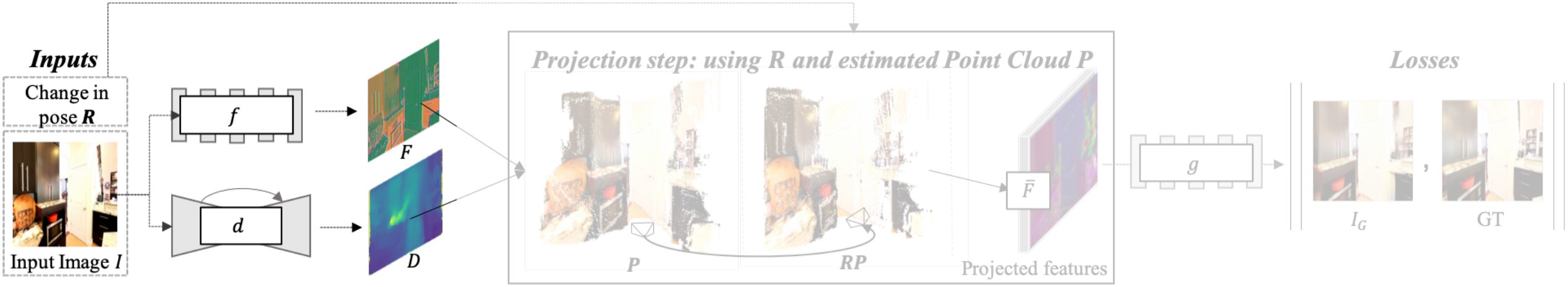
Learned 3D point cloud with trajectory overlaid

Generated views along the trajectory

Method

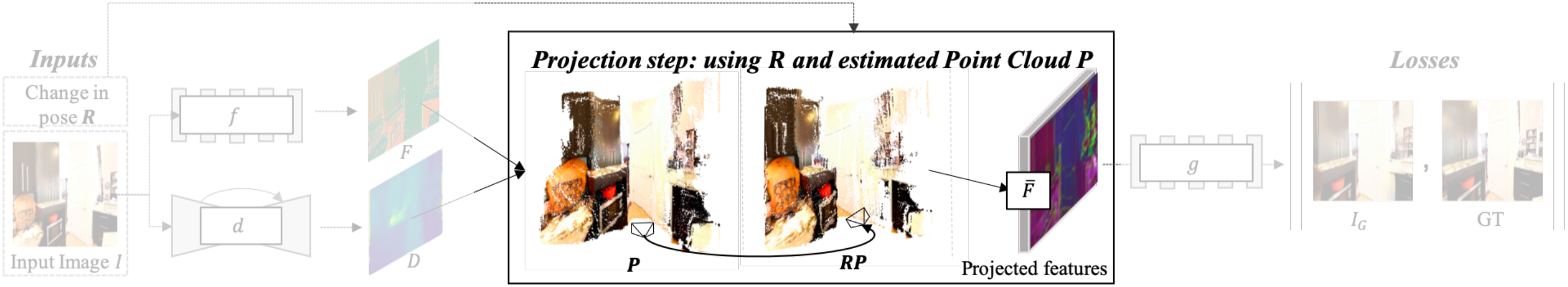

Inputs
Change in pose R
Input Image I
f
d
F
D
Projection step: using R and estimated Point Cloud P
P
RP
F̄
Projected features
g
Losses
I_G
GT

Method



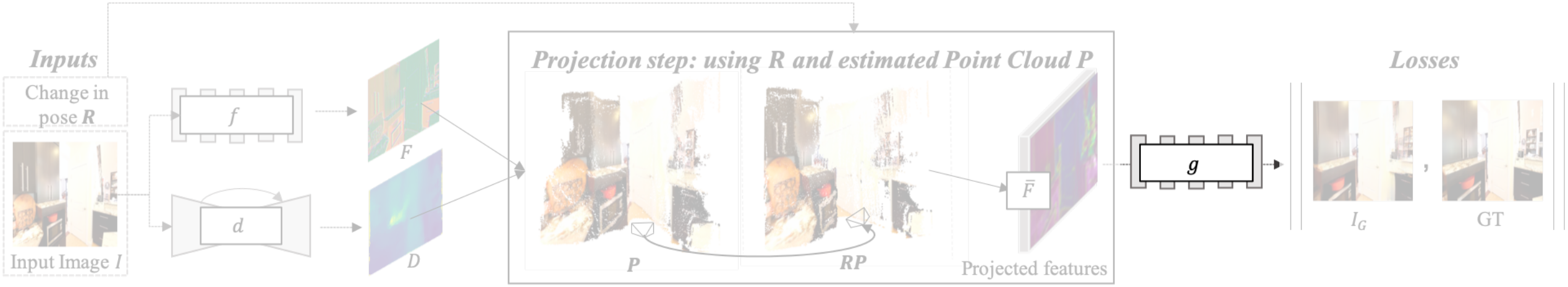
Input image I is embedded to a feature map F and a depth map D of the same input resolution as I

Method



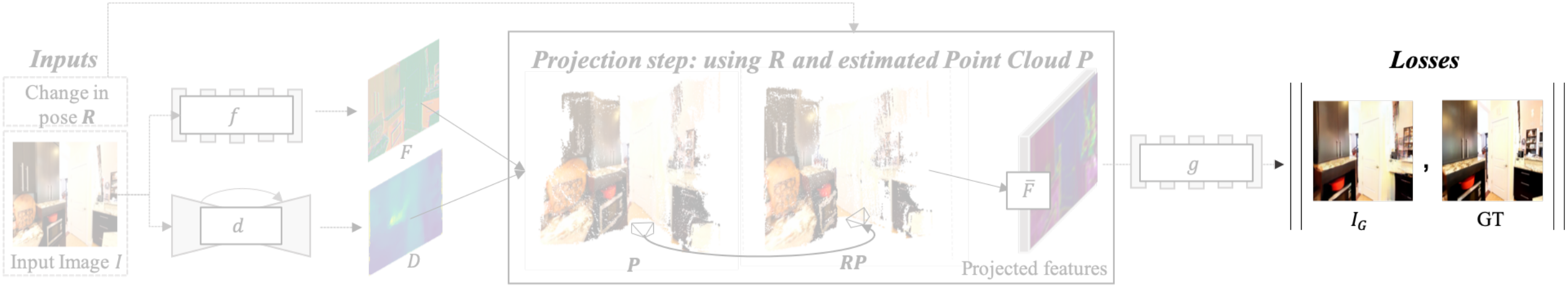
F and D generate a point cloud of feature vectors P. P is transformed via the given transformation R and rendered to the new view

Method



Even if features are projected in a geometrically accurate manner, the new feature map can still have holes (especially for large transforms R). The refinement module fills in the holes → **Inpainting**

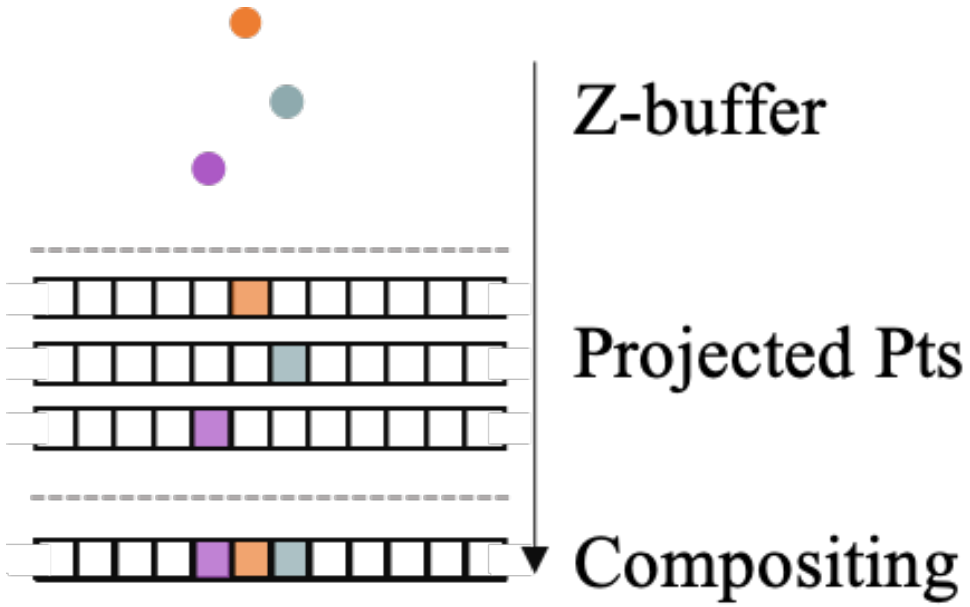
Method



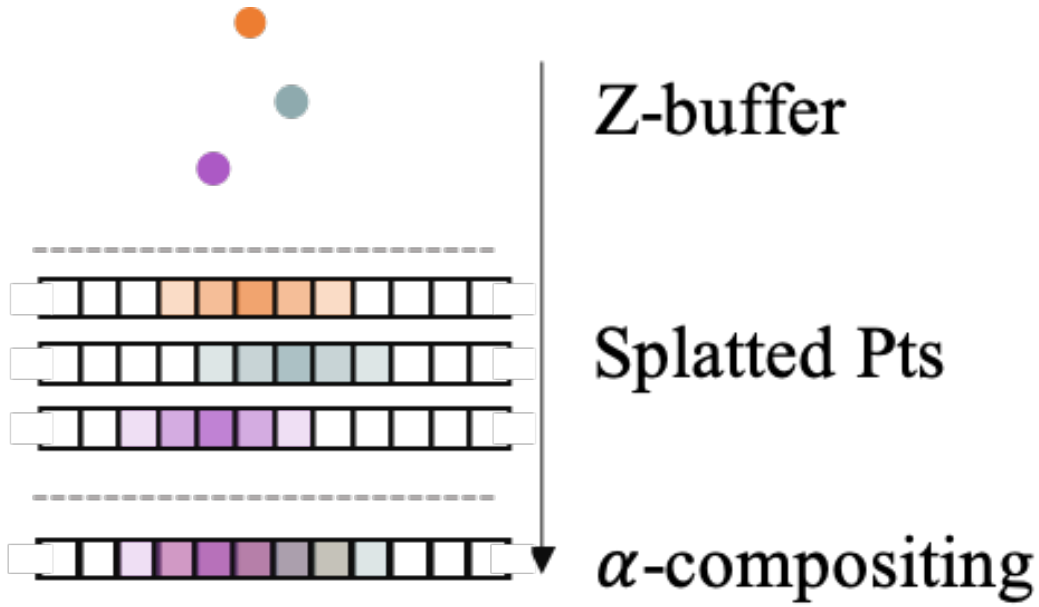
The loss is computed between the predicted view and the ground truth view. Plus a discriminator of course!

Differentiable Point Cloud Renderer

Forward pass:



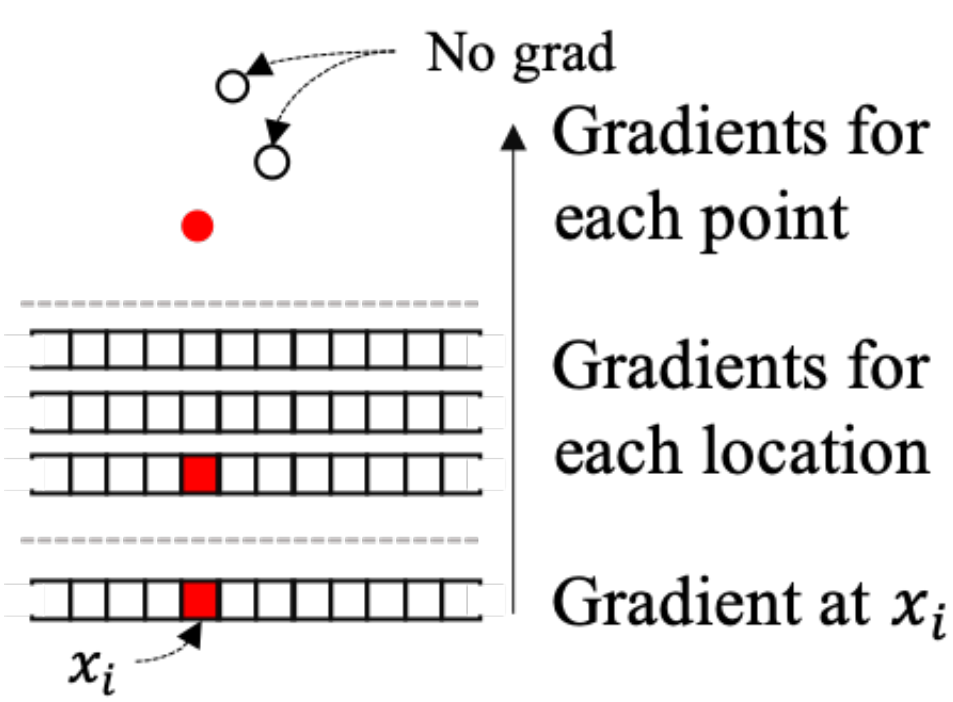
Naive



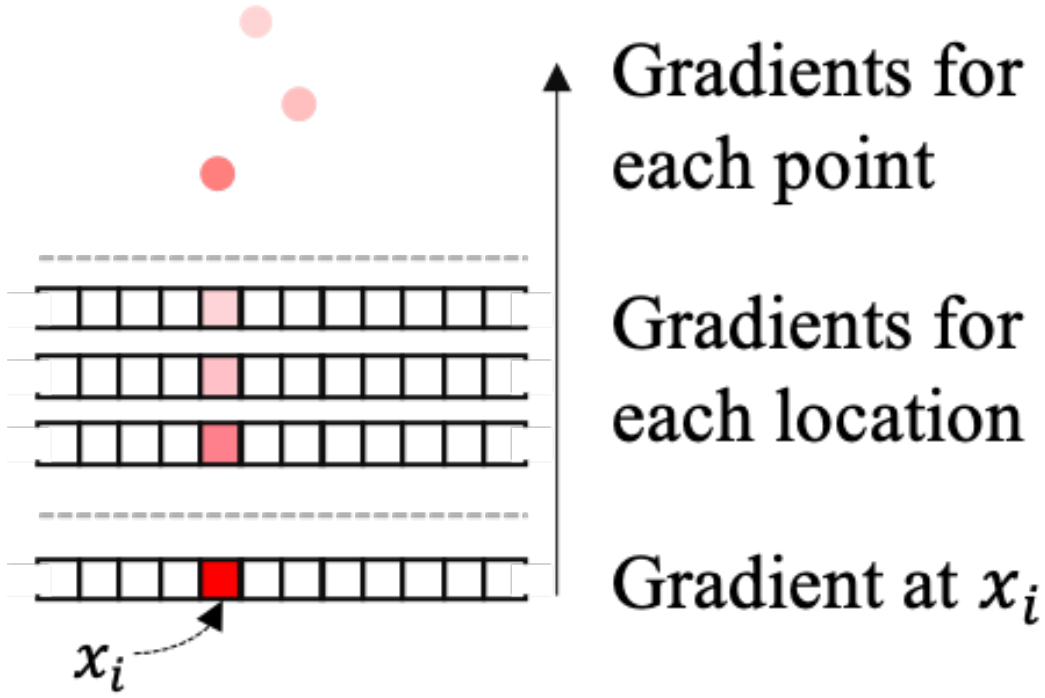
Ours

Differentiable Point Cloud Renderer

Backward pass:



Naive

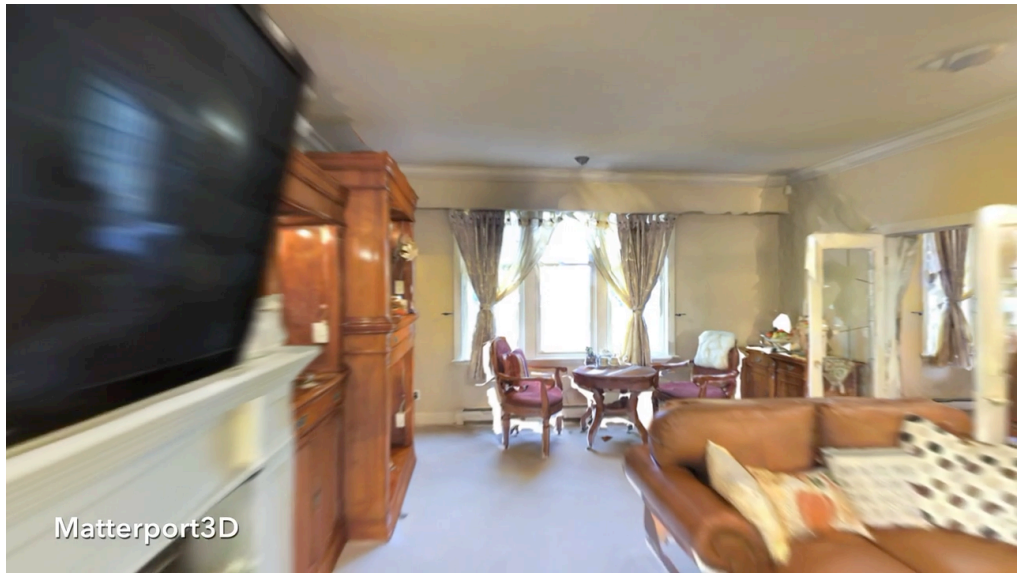


Ours

Datasets

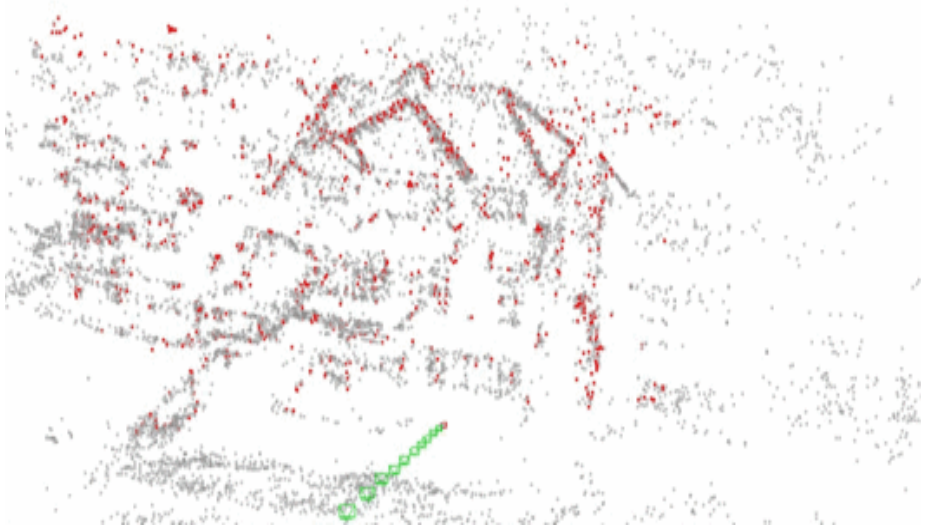
AI Habitat

Includes Matterport3D and Replica environments, which are scans of indoor scenes
[Savva et al. CVPR 2019]



RealEstate10k

Consists of video walkthroughs of properties
[Zhou et al. TOG 2018]



Experiments

	Matterport [4]									RealEstate10K [71]			Replica [55]		
	PSNR ↑			SSIM ↑			Perc Sim ↓			PSNR ↑	SSIM ↑	Perc Sim ↓	PSNR ↑	SSIM ↑	Perc Sim ↓
	Both	InVis	Vis	Both	InVis	Vis	Both	InVis	Vis						
1. SynSin (small ft)	21.36	20.37	22.06	0.72	0.70	0.70	1.58	0.43	0.91	$20.78_{3.51}$	$0.70_{0.15}$	$1.16_{0.53}$	21.64	0.79	1.70
2. SynSin (hard z)	20.14	19.51	20.62	0.66	0.68	0.64	1.93	0.46	1.20	$21.03_{4.01}$	$0.69_{0.16}$	$1.17_{0.58}$	21.95	0.79	1.69
3. SynSin (rgb)	21.03	19.98	21.69	0.68	0.69	0.66	2.15	0.47	1.35	$21.19_{4.30}$	$0.67_{0.15}$	$1.45_{0.54}$	21.71	**0.80**	2.03
4. SynSin	**21.82**	**20.59**	**22.63**	**0.73**	**0.71**	**0.71**	**1.51**	**0.42**	**0.86**	$\mathbf{22.78}_{5.25}$	$\mathbf{0.74}_{0.17}$	$\mathbf{0.95}_{0.61}$	**22.28**	**0.80**	**1.47**
5. SynSin (w/ GT)	23.76	20.84	26.87	0.82	0.75	0.84	1.22	0.47	0.55	–	–	–	24.84	0.88	1.08
6. SynSin (sup. by GT)	21.93	20.63	22.86	0.73	0.71	0.72	1.50	0.42	0.85	–	–	–	21.58	0.78	1.60
7. Im2Im	13.22	13.42	13.33	0.32	0.36	0.30	3.94	1.00	2.84	$16.51_{3.29}$	$0.47_{0.15}$	$1.94_{0.72}$	12.66	0.37	3.88
8. Vox w/ UNet	18.52	17.85	19.05	0.57	0.57	0.57	2.98	0.77	1.96	$17.31_{2.63}$	$0.53_{0.15}$	$2.30_{0.40}$	18.69	0.71	2.68
9. Vox w/ ours	20.62	19.64	21.22	0.70	0.69	0.68	1.97	0.47	1.19	$21.88_{4.39}$	$0.71_{0.15}$	$1.30_{0.55}$	19.77	0.75	2.24

Using a soft-K differentiable point cloud renderer improved the predicted results, both visually and for the metrics

Experiments

	Matterport [4]									RealEstate10K [71]			Replica [55]		
	PSNR ↑			SSIM ↑			Perc Sim ↓			PSNR ↑	SSIM ↑	Perc Sim ↓	PSNR ↑	SSIM ↑	Perc Sim ↓
	Both	InVis	Vis	Both	InVis	Vis	Both	InVis	Vis						
1. SynSin (small ft)	21.36	20.37	22.06	0.72	0.70	0.70	1.58	0.43	0.91	$20.78_{3.51}$	$0.70_{0.15}$	$1.16_{0.53}$	21.64	0.79	1.70
2. SynSin (hard z)	20.14	19.51	20.62	0.66	0.68	0.64	1.93	0.46	1.20	$21.03_{4.01}$	$0.69_{0.16}$	$1.17_{0.58}$	21.95	0.79	1.69
3. SynSin (rgb)	21.03	19.98	21.69	0.68	0.69	0.66	2.15	0.47	1.35	$21.19_{4.30}$	$0.67_{0.15}$	$1.45_{0.54}$	21.71	**0.80**	2.03
4. SynSin	**21.82**	**20.59**	**22.63**	**0.73**	**0.71**	**0.71**	**1.51**	**0.42**	**0.86**	$\mathbf{22.78}_{5.25}$	$\mathbf{0.74}_{0.17}$	$\mathbf{0.95}_{0.61}$	**22.28**	**0.80**	**1.47**
5. SynSin (w/ GT)	23.76	20.84	26.87	0.82	0.75	0.84	1.22	0.47	0.55	–	–	–	24.84	0.88	1.08
6. SynSin (sup. by GT)	21.93	20.63	22.86	0.73	0.71	0.72	1.50	0.42	0.85	–	–	–	21.58	0.78	1.60
7. Im2Im	13.22	13.42	13.33	0.32	0.36	0.30	3.94	1.00	2.84	$16.51_{3.29}$	$0.47_{0.15}$	$1.94_{0.72}$	12.66	0.37	3.88
8. Vox w/ UNet	18.52	17.85	19.05	0.57	0.57	0.57	2.98	0.77	1.96	$17.31_{2.63}$	$0.53_{0.15}$	$2.30_{0.40}$	18.69	0.71	2.68
9. Vox w/ ours	20.62	19.64	21.22	0.70	0.69	0.68	1.97	0.47	1.19	$21.88_{4.39}$	$0.71_{0.15}$	$1.30_{0.55}$	19.77	0.75	2.24

Training with ground truth depth, for the benchmarks for which they are available (not RealEstate10k) introduced only a small gain, suggesting that gt depth might not be crucial for this task.

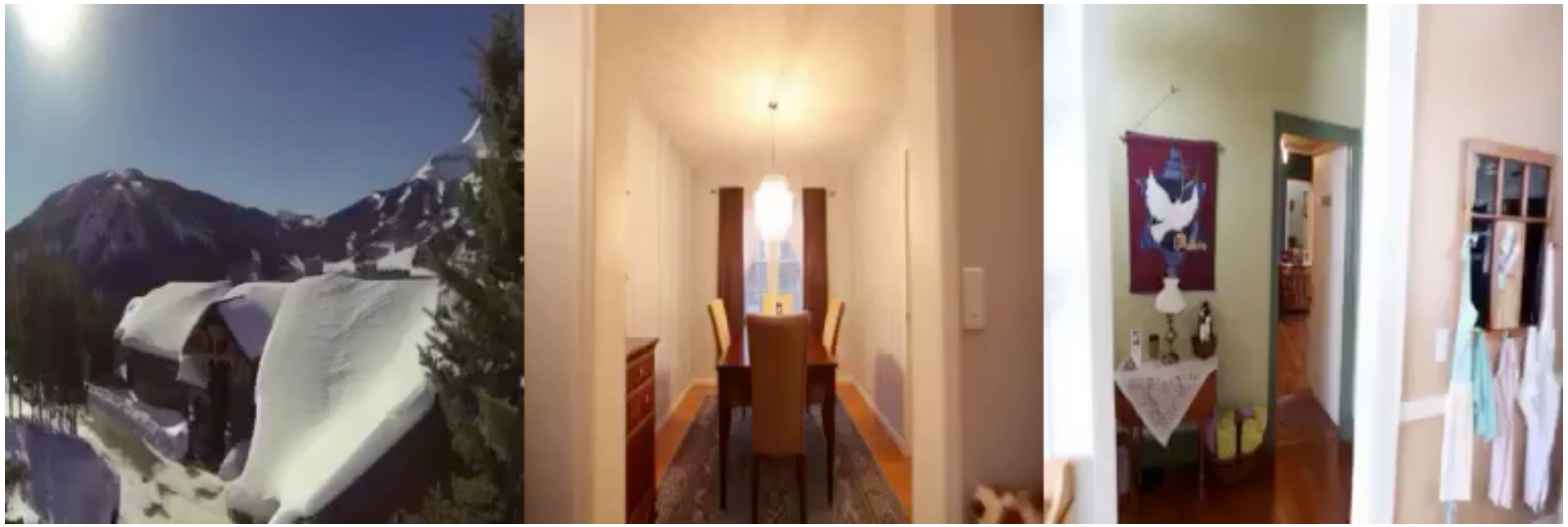
Experiments

	Matterport [4]									RealEstate10K [71]			Replica [55]		
	PSNR ↑			SSIM ↑			Perc Sim ↓			PSNR ↑	SSIM ↑	Perc Sim ↓	PSNR ↑	SSIM ↑	Perc Sim ↓
	Both	InVis	Vis	Both	InVis	Vis	Both	InVis	Vis						
1. SynSin (small ft)	21.36	20.37	22.06	0.72	0.70	0.70	1.58	0.43	0.91	$20.78_{3.51}$	$0.70_{0.15}$	$1.16_{0.53}$	21.64	0.79	1.70
2. SynSin (hard z)	20.14	19.51	20.62	0.66	0.68	0.64	1.93	0.46	1.20	$21.03_{4.01}$	$0.69_{0.16}$	$1.17_{0.58}$	21.95	0.79	1.69
3. SynSin (rgb)	21.03	19.98	21.69	0.68	0.69	0.66	2.15	0.47	1.35	$21.19_{4.30}$	$0.67_{0.15}$	$1.45_{0.54}$	21.71	**0.80**	2.03
4. SynSin	**21.82**	**20.59**	**22.63**	**0.73**	**0.71**	**0.71**	**1.51**	**0.42**	**0.86**	$\mathbf{22.78}_{5.25}$	$\mathbf{0.74}_{0.17}$	$\mathbf{0.95}_{0.61}$	**22.28**	**0.80**	**1.47**
5. SynSin (w/ GT)	23.76	20.84	26.87	0.82	0.75	0.84	1.22	0.47	0.55	–	–	–	24.84	0.88	1.08
6. SynSin (sup. by GT)	21.93	20.63	22.86	0.73	0.71	0.72	1.50	0.42	0.85	–	–	–	21.58	0.78	1.60
7. Im2Im	13.22	13.42	13.33	0.32	0.36	0.30	3.94	1.00	2.84	$16.51_{3.29}$	$0.47_{0.15}$	$1.94_{0.72}$	12.66	0.37	3.88
8. Vox w/ UNet	18.52	17.85	19.05	0.57	0.57	0.57	2.98	0.77	1.96	$17.31_{2.63}$	$0.53_{0.15}$	$2.30_{0.40}$	18.69	0.71	2.68
9. Vox w/ ours	20.62	19.64	21.22	0.70	0.69	0.68	1.97	0.47	1.19	$21.88_{4.39}$	$0.71_{0.15}$	$1.30_{0.55}$	19.77	0.75	2.24

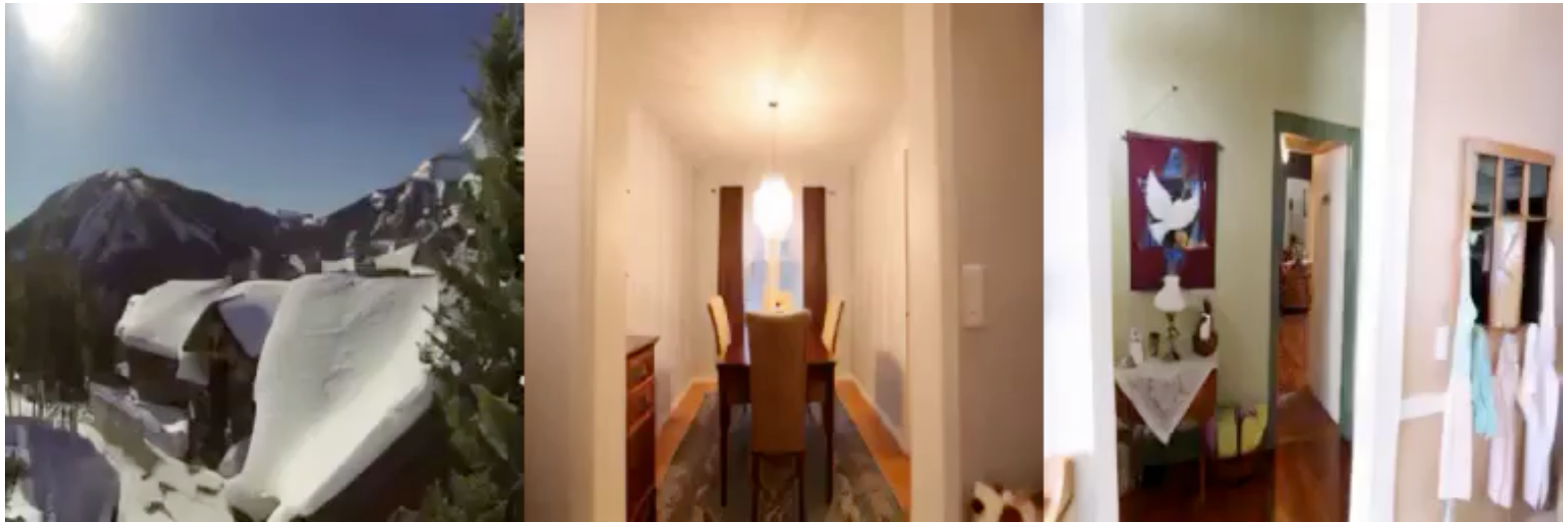
We compare to an image-to-image approach (no 3D) and a voxel-based approach, and found that at 3D point cloud representation performs significantly better, visually and for the metrics.

Results

Ours



Voxel-based



Results

Ours

Voxel-based

Generalization to higher resolutions

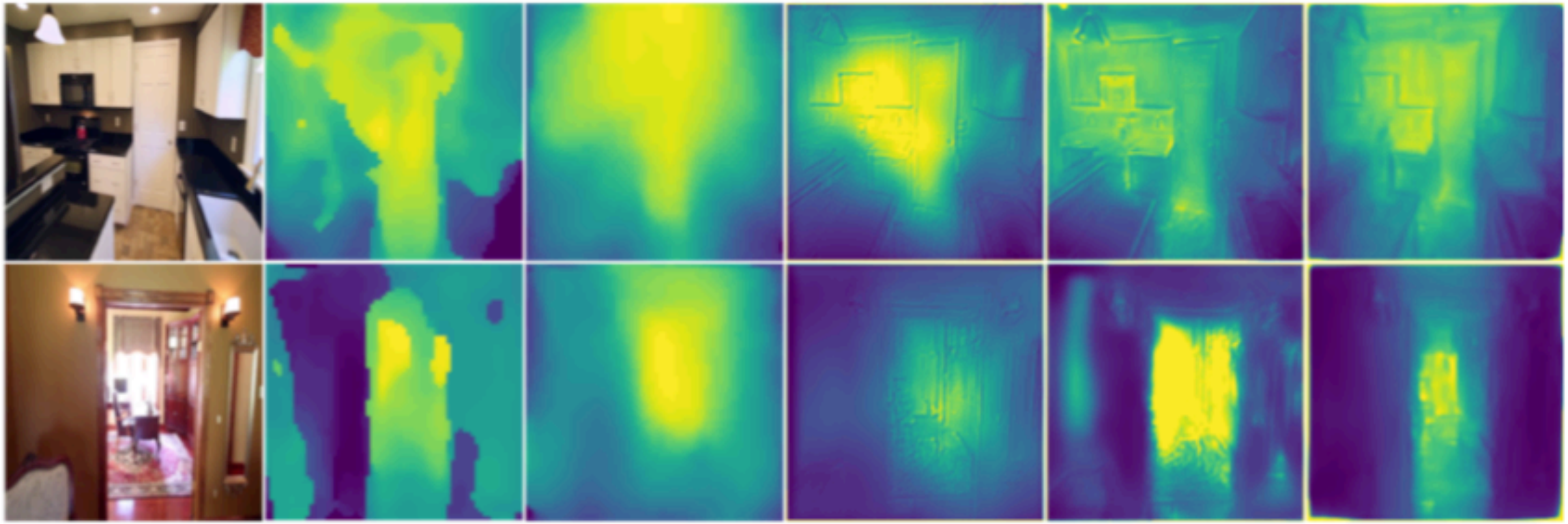
Input Image

Target Image

Vox w/ Ours

SynSin

What do the depth maps look like?



Thank you!